UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DESARROLLO DE MODELOS DE SISTEMAS DE INFORMACION

Pablo Gros Amorós
Tutor: Fernando Díez Rubio

ENERO 2017

Resumen

Este Trabajo Fin de Grado tiene como objeto el desarrollo (en su ciclo de vida completo de análisis, diseño, implementación y pruebas) de una serie de mejoras y funcionalidades añadidas sobre un sistema de gestión de información basado en modelos. El trabajo ha sido realizado en el ámbito de la empresa *B2T Concept*, dedicada al desarrollo de sistemas de información de una forma rápida y eficiente gracias a sus tecnologías de modelado.

Como parte del trabajo, se realiza un breve estudio de las tecnologías y metodologías propias de la Ingeniería de Modelos, y se explican de forma detallada los elementos y herramientas proporcionados por la empresa para la realización de las tareas. En este sentido ha sido necesario ampliar conocimientos desarrollados durante la titulación, ampliándolos con el estudio de métodos de desarrollo ágil de software, metodologías de desarrollo basadas en modelado, gestión de bases de datos, y despliegue de sistemas como servicios accesibles a través de Internet.

El proyecto sobre el que se ha trabajado es un sistema de ticketing, que permite gestionar y mantener de forma centralizada la lista de incidencias, peticiones y solicitudes de una compañía, por parte del servicio técnico (*helpdesk*). El sistema ha sido desarrollado en ciclos de iteración con el cliente, con lo que las actualizaciones se realizaban en periodos de tiempo muy cortos y siempre obteniendo *feedback* del usuario final.

A lo largo del documento se especifican las características del sistema, haciendo hincapié en las partes que tienen más relación con las tareas solicitadas. Se exponen de forma clara los requisitos especificados y el desarrollo de la solución para cada uno de ellos. Después se tratan las pruebas realizadas y las incidencias encontradas. Las funcionalidades añadidas han sido integradas en el sistema general y lanzadas a producción por lo que el sistema está funcionando de forma activa para el cliente, si bien el acceso de los usuarios es gradual dada la cantidad de información que se procesa.

El documento termina con una conclusión en la que se valoran los trabajos realizados en base a los objetivos planteados, y una reflexión sobre las metodologías basadas en modelado, indicando ventajas y desventajas del mismo. También se explica de forma breve el trabajo futuro que se realizará de cara a este sistema.

Palabras clave

Ingeniería basada en modelos (MDE), Sistemas de Información, Desarrollo rápido de aplicaciones

Abstract

This final project is aimed at the development (in its complete life cycle analysis, design, implementation and testing) of a number of improvements and added functionality to an information system based on models. The work has been realized in the company *B2T Concept*, dedicated to the development of information systems quickly and efficiently thanks to its modeling technologies.

As part of the work, a brief study of technologies and own model engineering methodologies is performed, and explained in detail the elements and tools provided by the company to carry out the tasks. It has been necessary to expand knowledge developed during the titulation, extending the study of agile software development, development methodologies based on modeling, management databases, and deployment of systems and services accessible via the Internet.

The project on which work has been done is a ticketing system that lets you manage and maintain centralized list of incidents, requests and applications from a company, by the technical service (helpdesk). The system has been developed in iteration cycles with the client, updates were made in very short periods of time and always getting feedback from the end user.

Throughout the document the system characteristics are specified, with emphasis on the parts that are more related to the tasks requested. Clearly exposed the specified requirements and the development of the solution for each of them. Afterwards, tests and incidents encountered are addressed. The added features have been integrated into the overall system and launched into production environment so the system is actively working for the customer, while user access is gradual given the amount of information processed.

The paper ends with a conclusion of the work carried out on the basis of the objectives, and a reflection on the modeling methodologies, indicating advantages and disadvantages of it. It also explains briefly the future work to be done ahead of this system.

Keywords (inglés)

Model Driven Engineering (MDE), Information System, Agile Software Development

Agradecimientos

Agradezco a mis compañeros de B2T por acogerme en el equipo, enseñarme y formarme, y ayudarme en todo momento con las dificultades que iban surgiendo. A Pablo por ser mi primer tutor, y Alfonso por coger el relevo. A Víctor y José por compartir el estado de becarios y ser los primeros en ofrecerse para echar una mano.

No puedo olvidar a todos los compañeros y amigos que me han acompañado a lo largo de la carrera, los que empezaron y no acabaron, los que empezaron y acabaron a su debido momento, y los que aún siguen en contacto fuera de la facultad.

Tengo que agradecer el haber llegado hasta este momento sobre todo a mi familia, en concreto a mis padres, que han compartido conmigo toda esta etapa, aguantando mis cabezonerías y siempre animándome para continuar hacia delante sin que me pueda el esfuerzo.

A todos, gracias.

Pablo Gros Amorós
Enero 2017

INDICE DE CONTENIDOS

1 INTRODUCCIÓN 1

1.1 MOTIVACIÓN 1
1.1 CONTEXTO 1
Model Driven Engineering (MDE) 2
1.2 ORGANIZACIÓN DE LA MEMORIA 4

2 DESCRIPCIÓN DEL PROYECTO 5

2.1 DESCRIPCIÓN DEL PROBLEMA 5
2.2 OBJETIVOS 5

3 TECNOLOGÍAS Y METODOLOGÍA DE DESARROLLO 7

3.1 ARQUITECTURA DE LOS SISTEMAS 7
3.2 SCOOP 8
3.3 SCOOP BPME 8
3.4 SQL SERVER 8
3.5 DEVR + JCOR 9
3.6 SVN 9
3.7 METODOLOGÍA DE DESARROLLO 9

4 ANÁLISIS 13

4.1 REQUISITOS GENERALES DEL SISTEMA 13
4.2 REQUISITOS ESPECÍFICOS 14
General 14
Ticket de incidencias 14
Ticket de cambio de parámetros 16
Ticket de cambios 17
Ticket de peticiones 18

5 DISEÑO 19

5.1 CASOS DE USO 19
5.2 MODELO DE PROCESOS 20
BPM - Cambio de parámetros 20
BPM – Cambios 21
BPM – Peticiones 21
5.3 MODELO CONCEPTUAL 23
Incidencia (ReqIncident) 23
Cambio de parámetros (ReqChangeParams) 23
Solicitud de cambios (ReqChange) 24
Petición (Request) 25
Tareas del ticket de cambios (TaskITSM) 26
Grupos aprobadores/revisores (RelGrupoChange) 26
Servidores e instancias (RelReqChgInstance/RelReqChgServer) 26
Tipologías o listas administrables 26
5.4 MODELO DE DATOS 27
5.4.1 ITS_CHANGE_PARAM 27
5.4.2 ITS_RELGRUPOCHANGE 28
5.4.3 ITS_RELREQCHGSERVER 29
5.4.4 RS_GRUPO_TRABAJO 29
5.5 MODELO DE INTERFACES 30
5.6 MODELO DE SEGURIDAD 31
5.6.1 Headers o Bandejas 31
5.6.2 Perfiles 32

6 DESARROLLO 34

7 INTEGRACIÓN, PRUEBAS Y RESULTADOS 47

8 CONCLUSIONES Y TRABAJO FUTURO 49

8.1 CONCLUSIONES 49
8.2 TRABAJO FUTURO 49

REFERENCIAS 51

GLOSARIO 51

ANEXO A 52

INDICE DE FIGURAS

FIGURA 1: ARQUITECTURA DE LOS ISTEMAS……6

FIGURA 2: DESARROLLO EN SPIRAL……8

FIGURA 3: CICLO DE ESARROLLO……10

FIGURA 4: CASOS DE SO……17

FIGURA 5: ETAPAS DEL FLUJO DE AMBIOS……19

FIGURA 6: ESTRUCTURA DE UNA NTERFAZ……27

FIGURA 7: PANTALLA PRINCIPAL DE UN ÉCNICO……29

FIGURA 8: CONSULTA DE UN GRUPO DE RABAJO……30

FIGURA 9: CONSULTA DE UNA NCIDENCIA……31

FIGURA 10: FORMULARIO DE LINK MASIVO DE NCIDENCIAS……33

FIGURA 11: CONSULTA EN EL CAMBIO DE TAREAS Y RUPOS……38

FIGURA 12: FORMULARIO DE ALTA DE UNA TAREA……39

FIGURA 13: INTERFAZ DE ASOCIACIÓN DE GRUPOS PROBADORES……40

INDICE DE TABLAS

TABLA 1: MODELO DE ESTADOS DE NCIDENCIAS……20

TABLA 2: MODELO DE ESTADOS DE CAMBIO DE ARÁMETROS……21

TABLA 3: MODELO DE ESTADOS DE AMBIOS……21

TABLA 4: MODELO DE ESTADOS DE ETICIÓN……22

TABLA 5: TABLA DE CAMBIO DE ARÁMETROS……24

TABLA 6: TABLA DE RELACIÓN CAMBIO – GRUPO RABAJO……25

TABLA 7: TABLA DE RELACIÓN CAMBIO – ERVIDOR……26

TABLA 8: TABLA DE GRUPO DE RABAJO……26

1 Introducción

1.1 Motivación

Este trabajo fin de grado forma parte de las actividades desarrolladas como parte del proyecto realizado en la empresa *B2T Concept* para la implementación de un sistema de ticketing y seguimiento de incidencias.

El problema que presenta el cliente se puede desglosar en dos partes: primero, la necesidad de actualizar los sistemas antiguos presentes en la empresa que se han quedado obsoletos, o bien no contemplan los métodos de trabajo que se quieren implantar, ni refleja los nuevos procesos internos; segundo, el requerimiento de una herramienta que abarque todos los procesos que presenta la compañía en un mismo entorno, evitando el uso de diferentes programas. Así, surge la oportunidad de desarrollar un sistema informático general tanto para el departamento de tecnología como para el usuario corriente, que contará con varios módulos de diferente funcionalidad, ya sea gestión de la demanda, gestión de proyectos o un sistema de ticketing, del que trata este trabajo de fin de grado.

Principalmente son dos las razones que han motivado la realización del trabajo. Por una parte, poder desarrollar el mismo en el contexto de la Ingeniería Basada en Modelos, con los paradigmas, metodologías, ventajas y desventajas que ello supone. En el caso específico de *B2T Concept*, la idea final subyacente es la de desarrollar Sistemas de Información basados en Modelado de procesos **sin tener que programar,** en el sentido clásico del término. Si bien esto último es difícil (siempre hay que escribir algún algoritmo de funcionalidad muy específica para responder a las necesidades del cliente), es cierto que la combinación entre la concepción de los proyectos/sistemas y las herramientas adecuadas, véase entornos de desarrollo de modelos, motores que interpretan los modelos y los transforman a texto, etc. ayuda mucho a éste propósito, facilitando las tareas de desarrollo, y permitiendo una mejor solución ya que se ha podido diseñar mejor y se ha consumido menos tiempo en la programación.

Por otra parte, otra de las motivaciones del trabajo ha sido la oportunidad de desarrollar un proyecto en un ambiente empresarial, adquiriendo cierta experiencia y pudiendo observar y apreciar el desarrollo de proyectos de cara a clientes reales, con los aspectos de responsabilidad, gestión de tareas y plazos, etc. que ello conlleva.

1.1 Contexto

No descubrimos ninguna novedad al destacar el impacto de las TIC en el mundo empresarial actual. Sin embargo, la velocidad a la que cambian y surgen nuevas técnicas hace que las empresas tengan una creciente necesidad de adaptarse a los nuevos métodos de gestión de la información para mantenerse competitivas. No sólo se manejan transacciones o intercambios de información, también es necesario tomar el control de los procesos de negocio [1].

En este sentido, la empresa *B2T Concept* trabaja con la idea de crear sistemas de gestión de la información que ayuden a las empresas a funcionar de una manera más organizada y

eficiente, ofreciendo además un tiempo de desarrollo ajustado y la posibilidad de mantener actualizados los sistemas con los cambios que puedan surgir a medida que evoluciona el negocio. Esto se consigue gracias a tecnología basada en modelos y meta-modelos, y en la forma en la que éstos representan el negocio.

Una parte fundamental en la realización de este trabajo ha estado ligada a las técnicas de modelado, que describimos a continuación.

Model Driven Engineering (MDE)

El uso de metodologías de modelado resulta altamente beneficioso y se está consolidando, en los últimos años, debido a la complejidad en el diseño y desarrollo de los sistemas de información. De hecho, en el desarrollo de aplicaciones software, se ha experimentado una clara tendencia a pasar de aproximaciones centradas en el código a aproximaciones centradas en modelos.

Estas metodologías se basan en la utilización de lenguajes de modelado que permitan una completa descripción de dichos sistemas antes de su construcción, posibilitando la detección temprana de errores en fase de diseño. El Desarrollo Basado en Modelos (*MDD, Model-Driven Development*) es un paradigma emergente que permite resolver muchos de los problemas asociados con la composición e integración de grandes sistemas. Se basa en el uso de modelos para representar los elementos de un sistema y sus relaciones. Estos modelos se emplean como entradas y salidas de todos los estados del ciclo de desarrollo del sistema hasta su generación.

En este contexto, la Ingeniería Basada en Modelos (*MDE o Model-Driven Engineering*) es una metodología que promueve el uso intensivo de modelos para hacer frente a la complejidad inherente al diseño de aplicaciones. MDE va más allá que las simples actividades de desarrollo, de hecho, combina otras tecnologías como por ejemplo el uso de lenguajes de modelado específicos de dominio, así como generadores o transformadores.

En definitiva, por un lado, los lenguajes específicos de dominio se basan en meta-modelos, permitiendo la descripción de la estructura, comportamiento y requisitos de un sistema. Por otro lado, las transformaciones de modelos permiten el análisis de los modelos para la obtención bien de código (transformaciones modelo-texto) bien de otros modelos (transformaciones modelo-modelo). Se consigue, por lo tanto, reducir los costes y tiempos de desarrollo de las aplicaciones, aumentando su fiabilidad, flexibilidad y reutilización, además de facilitar la descripción de nuevas arquitecturas de reconfiguración [2].

Haciendo uso de estas tecnologías, el trabajo llevado a cabo ha consistido en el desarrollo de los elementos que conforman el sistema de tickets. Todas las implementaciones llevadas a cabo se han efectuado en periodos de tiempo muy cortos y siempre obteniendo *feedback* del usuario final, adaptando los desarrollos a los requisitos que este iba especificando.

En resumen, el producto generado se trata de un sistema formado por cuatro módulos:

- **Incidencias:** Una incidencia es cualquier problema ocurrido con una aplicación de la empresa o alguno de los servicios ofrecidos a los usuarios. Generalmente cuando un usuario se encuentra con un mal funcionamiento de alguna de las herramientas de trabajo y tiene que contactar con el departamento técnico se le solicita que abra una incidencia, para que los técnicos puedan recibir la información del problema,

encolarlo en base a la prioridad/urgencia del mismo, y disponer de un medio de comunicación con el solicitante.

- **Cambio de parámetros:** La gestión de parámetros es un elemento crítico dentro de la configuración de las aplicaciones de negocio. Un parámetro de control es un valor que puede modificarse dentro de una aplicación y que tiene un impacto inmediato en la forma o en el tratamiento de las cuentas de cliente. Los parámetros son utilizados dentro de las aplicaciones para configurar ciertas características de las aplicaciones sin requerir un esfuerzo adicional de desarrollo. En ocasiones es necesario efectuar modificaciones sobre los parámetros. Estas solicitudes son requeridas desde el área de negocio siendo responsabilidad del propietario del proceso de parámetros asegurar la correcta implementación y verificación del cambio.
- **Cambios:** Un cambio es similar a un cambio de parámetros, pero no está limitado a un valor concreto de una aplicación, sino que aplica directamente al producto software y sus infraestructuras relacionadas (servidores, instancias). Se realiza una solicitud de este tipo cuando se va a instalar una nueva versión de una herramienta, y es necesario documentar el proceso técnico completo, los elementos a los que afecta, y el periodo en el que se efectuará dicho cambio, ya que generalmente implica un parón de las aplicaciones involucradas que afecte a la actividad de los usuarios. Un cambio también lleva asociado un plan de pruebas para antes y después de la implementación del mismo, y un detalle de restauración en caso de que se produzca un fallo. Tanto los cambios de parámetros como los cambios están enfocados a un usuario técnico.
- **Peticiones:** Las peticiones son solicitudes de los usuarios que necesitan disponer de una aplicación o un servicio en un momento determinado. Cuando un empleado necesita instalar alguna herramienta tiene que realizar una petición y ser aceptada para su posterior instalación. Generalmente estas peticiones se hacen con programas y herramientas, pero también envuelve elementos de infraestructura, como pueden ser dispositivos hardware (monitores y periféricos, ordenadores…), servidores, o incluso cableado o sistemas de aire acondicionado. Estas peticiones están orientadas a un usuario medio, que no pertenece al departamento técnico.

A lo largo del documento se especifican las características del sistema, haciendo hincapié en las partes que tienen más relación con las tareas solicitadas. Se exponen de forma clara los requisitos especificados y el desarrollo de la solución para cada uno de ellos. Después se tratan las pruebas realizadas y las incidencias encontradas. Las funcionalidades añadidas han sido integradas en el sistema general y lanzadas a producción por lo que el sistema está funcionando de forma activa para el cliente, si bien el acceso de los usuarios es gradual dada la cantidad de información que se procesa.

El documento termina con una conclusión en la que se valoran los trabajos realizados en base a los objetivos planteados, y una reflexión sobre las metodologías basadas en modelado, indicando ventajas y desventajas del mismo. También se explica de forma breve el trabajo futuro que se realizará de cara a este sistema.

1.2 Organización de la memoria

La memoria consta de los siguientes capítulos:

i) Introducción: en esta sección se introduce de forma general al sistema sobre el que ha trabajado el alumno, así como una breve contextualización al mundo de la ingeniería de modelos en la que se basa la metodología de trabajo de la empresa.

ii) Descripción del proyecto: en este apartado se proporciona una visión general del sistema que se ha desarrollado, así como los objetivos que se persiguen con la implementación y que se han tenido en cuenta a la hora de realizar este Trabajo Fin de Grado.

iii) Tecnologías y metodologías de trabajo: donde se detallan las diferentes herramientas empleadas para la realización de las tareas y el cumplimiento de los requisitos, así como la arquitectura del sistema.

iv) Análisis: sección que recoge de forma exhaustiva las especificaciones extraídas del cliente.

v) Diseño: en este apartado se explica de forma detallada el diseño general del sistema, haciendo énfasis en las partes que más papel han jugado en el desarrollo de este trabajo y que fueron desarrolladas o modificadas por el alumno. Se expondrán de forma clara los distintos tipos de modelos empleados y los conceptos asociados, además de las especificaciones de las bases de datos empleadas.

vi) Desarrollo: en este capítulo se expone de manera detallada la implementación de los requisitos definidos en el apartado 4. Se tratarán los casos más representativos para dar un ejemplo de las diversas aproximaciones tomadas.

vii) Integración, pruebas y resultados: en este apartado se hablará de cómo se han ensamblado las piezas para formar el sistema y las pruebas realizadas sobre el mismo.

viii) Conclusiones y trabajo futuro: se expondrán las conclusiones sobre el trabajo realizado, se hará una pequeña reflexión sobre los métodos de desarrollo conocidos por el estudiante indicando ventajas y desventajas sobre la los basados en modelos, y se indicarán las tareas futuras que se realizarán continuando el trabajo realizado.

2 Descripción del proyecto

El propósito de este Trabajo de Fin de Grado es diseñar y desarrollar una herramienta software a medida de gestión de tickets y seguimiento. En este apartado se incluye una descripción del problema y los objetivos que se pretenden cumplir con la implementación.

2.1 Descripción del problema

IT Service Management (ITSM) es un sistema de gestión de tickets que surge como solución a la necesidad de la compañía de centralizar los distintos tipos de ticket que recibe el departamento de asistencia técnica, permitiendo de una forma sencilla realizar peticiones por parte de los usuarios, y realizar un seguimiento y control de las tareas por parte de los técnicos. Los tickets están apoyados sobre una fuente de aplicaciones y servicios del cliente que se encuentran clasificados en varios niveles de categorías y subcategorías. Este módulo se denomina CMDB, y es un subsistema de clasificación y gestión de aplicaciones basado en componentes, a modo de base de datos de todo el software, servidores, y aplicaciones de la empresa.

La gestión de los tickets tiene las siguientes características:

- Hay distintos tipos de tickets: Incidencias, peticiones, cambios, parámetros y problema (éste último no ha sido especificado aún en el momento de realizar el trabajo). Cada tipo de ticket tiene su formulario descriptivo distinto. Se diferenciarán los mecanismos de clasificación de tickets en apertura y en cierre, para facilitar la obtención de estadísticas y poder realizar informes.
- Los tickets están integrados con la CMDB. La integración se puede realizar a distintos niveles para facilitar la entrada y clasificación de los datos. Por ejemplo, los usuarios que incorporan Incidencias de aplicación pueden ver solo el nivel 1 de aplicaciones en la CMDB para catalogación del ticket.
- Flujos de trabajo. El sistema estará basado en workflows configurables para cada tipo de ticket (cada flujo será especificado en el apartado 5).
- Usuario. Hay tres tipos de usuarios en el sistema: solicitantes, que realizan peticiones únicamente; técnicos, encargados de resolver peticiones, pero que también pueden solicitarlas; administrador/es, que se encargan de la configuración y parametrización de ciertos elementos del sistema. Los tickets son asignados a grupos de usuarios, denominados grupos de trabajo, y que estarán encargados de la resolución del mismo.

2.2 Objetivos

El propósito de las tareas a desarrollar como parte de este Trabajo de Fin de Grado son añadir mejoras y nuevas funcionalidades al sistema de tickets utilizando tecnología basada en modelos. En particular hemos agrupado los objetivos en los siguientes grupos:

- **Correcciones y mejoras solicitadas por el cliente sobre el módulo de Incidencias**. El ticket de incidencias fue el primero que se implementó y se mostró al cliente para poder sentar una base sobre la que desarrollar el resto de tickets, en términos de interfaz y funcionalidad. Tras instalar el módulo en un entorno en el que el cliente puede realizar pruebas, obtuvimos el *feedback* y se solicitaron mejoras en términos de usabilidad y consulta de la información.

- **Análisis, diseño e implementación del módulo de Cambio de parámetros, Cambios y Peticiones.** El objetivo es ampliar las características del sistema de tickets desarrollando una solución software a medida del cliente. Para cada módulo se proporcionará un formulario de alta específico, donde se introducen los datos concretos de cada ticket, se definirá el proceso que sigue el ticket en base al procedimiento interno del cliente, y por último se proporcionará una interfaz cómoda para la consulta de la información, en forma de listados clasificados por ticket, y pantallas de visualización con acceso directo a los datos relacionados relevantes, para proporcionar una visualización global *a distancia de un click*.

3 Tecnologías y metodología de desarrollo

En este apartado se describirá de forma breve la metodología de desarrollo que se aplica a los proyectos, la arquitectura de los sistemas, y de forma más detallada las herramientas utilizadas durante el desarrollo.

3.1 Arquitectura de los sistemas

El siguiente diagrama representa la arquitectura de los sistemas desarrollados con la plataforma ScooP (herramienta explicada en el siguiente apartado) de desarrollo de modelos.

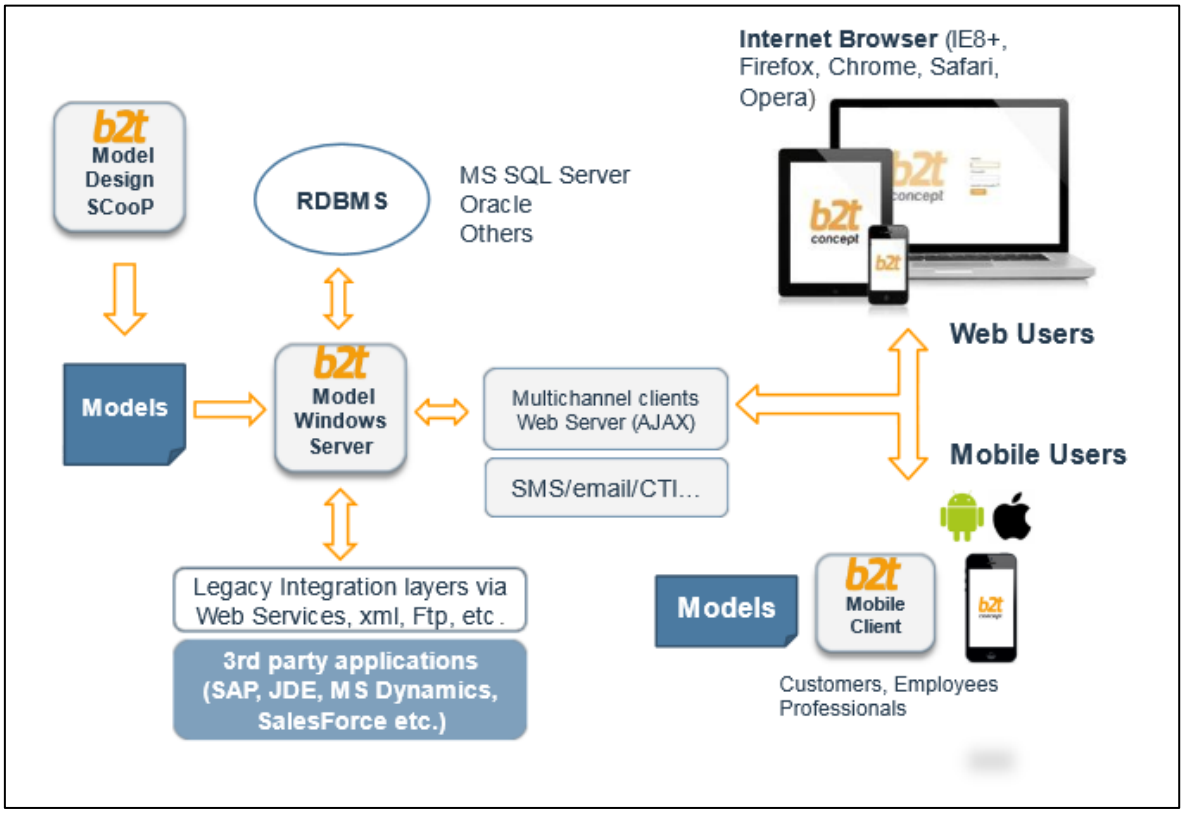

Figura 1: Arquitectura de los sistemas

Los componentes principales de la plataforma son:

- ScooP: como herramienta de diseño de modelos, ejecutado en los servidores propios de la empresa.
- ScooP Enterprise (Motor E3): que interpreta los modelos en base a un repositorio.
- Bases de datos: de tipo Oracle o MS SQL, por medio de un ODBC.
- ScooP Desktop (SABLE): cliente web basado en tecnología AJAX con HTML5, que genera interfaces web de usuario. Soporte para navegadores estándar.
- ScooP Mobile (ME2 Mobile): que utiliza HTML5 para las interfaces de usuario y Servicios Web para el intercambio de datos.

3.2 ScooP

ScooP es una herramienta de desarrollo rápido de aplicaciones basada en tecnología orientada a modelos. Permite la definición de modelos de negocio, su manipulación, y su transformación para obtener prototipos ejecutables. Diferencia entre Modelos de Procesos, dirigidos a modeladores funcionales y/o consultores que utilizan los modelos de forma descriptiva, y los Modelos Técnicos, creados por los técnicos.

Este trabajo se centra en este último tipo de modelos, que agrupan de forma genérica el modelo **conceptual**, los modelos de **interfaces**, los modelos de **seguridad** y los modelos de **datos**. Todos estos modelos los desarrollaremos de forma detallada en la siguiente sección.

Una vez definidos los diferentes modelos, ScooP permite generar los ficheros de clases y funciones que serán interpretados por el motor, dando lugar como resultado de la interpretación a una aplicación ejecutable.

Estos ficheros son tratados por el desarrollador para implementar los algoritmos que no pueden ser descritos en el modelo.

3.3 ScooP BPME

ScooP BPME es el motor de modelado de procesos de ScooP. Las dos funcionalidades primarias de SCooP BPMe son la definición de procesos de negocio y la definición de eventos.

Un proceso de negocio se compone principalmente de un conjunto de tareas que pueden llevarse a cabo (decisiones por parte de usuarios, tareas ejecutadas automáticamente, tareas de control del proceso...) y de un flujo que guía la ejecución de dichas tareas. Los procesos de negocio se ejecutan en el contexto de una aplicación y, dentro de ésta, en el contexto de un elemento de negocio concreto, con las consecuentes variaciones en el flujo y los resultados específicos de las tareas. En este sentido, el proceso de negocio representa un esquema de ejecución de un flujo.

Por el otro lado, un evento corresponde no sólo al concepto genérico de “evento” (un suceso) sino a una ejecución asociada a dicho suceso, especificada en el entorno de definición en base a una serie de opciones estándar. La finalidad de los eventos es la especificación prediseñada de ejecuciones asociadas a estos sucesos. Por ejemplo, se desea que, cada vez que se produzca una creación de un nuevo ticket en el sistema, se envíe un mail de aviso a una determinada dirección de correo.

3.4 SQL Server

Microsoft SQL Server es un gestor de bases de datos relacional. Como servidor de base de datos, permite las funciones de almacenamiento y recuperación de datos solicitados por otras aplicaciones o servicios a través de internet.

Mediante la definición de un ODBC (Open DataBase Connectivity) para el sistema, podemos acceder a los datos desde las aplicaciones generadas por ScooP.

3.5 DEVR + Jcor

La empresa B2T dispone de un lenguaje de programación propio denominado JCor, con el que se desarrolla todo el código. JCor es un lenguaje con sintaxis muy similar a JavaScript, interpretado y de orientación a objetos, enfocado a sistemas de gestión y con primitivas específicas para el tratamiento de modelos que se hace en la empresa. Los ficheros de clases y funciones generados automáticamente por ScooP están escritos en este lenguaje.

Para tratar con estos tipos de ficheros y disponer de un entorno dedicado al lenguaje, se dispone de la herramienta DEVR, también de desarrollo propio de la empresa. DEVR es un entorno de desarrollo que ayuda a la programación en lenguaje JCor, con repositorio de ayuda, debbuger, y asociación de ficheros de configuración (.ini). También cuenta con el motor que interpretará el código y permite lanzar las aplicaciones.

3.6 SVN

Subversion es la herramienta elegida para el control de versiones de los proyectos. Está sincronizado con la herramienta ScooP, de forma que los cambios producidos de forma local y subidos al repositorio puedan luego verse reflejados en el gestor de modelos para interactuar con ellos.

3.7 Metodología de desarrollo

El desarrollo en espiral (Figura 2) es la metodología ágil embebida en el ciclo de vida de ScooP que se utiliza para realizar la transformación del conocimiento empresarial en aplicaciones de negocio. Esta metodología está basada en un ciclo que podemos agrupar en 4 pasos, que se repiten de forma continua acortando las distancias entre la solución y las necesidades reales del negocio. Permite conocer la opinión del cliente en cada iteración haciendo que los requisitos y soluciones evolucionen minimizando los riesgos del desarrollo en cortos periodos de tiempo.

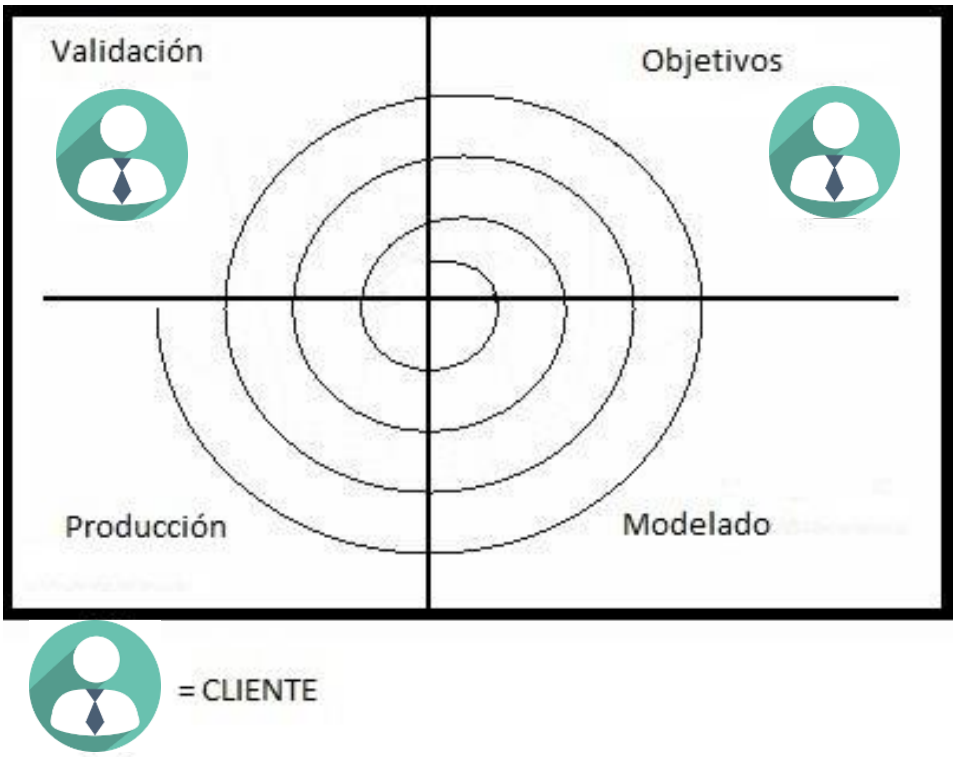

Figura 2: Desarrollo en espiral

Cada iteración consta de las siguientes fases:

1. **Objetivos - Comprensión del problema**. El problema de la empresa nace en el lado del negocio, los expertos en el problema son las personas dentro de la empresa cliente. Su conocimiento es por lo general una mezcla de conocimiento tácito y explícito, y las áreas de incertidumbre se ocultan con frecuencia. Utilizamos un enfoque centrado en el proceso para descomponer el escenario de negocios en los procesos, subprocesos, los conceptos y requisitos de las especificaciones que será la clave para entender el problema de la empresa, e identificar las inconsistencias y los riesgos. En resumen, llegar a un acuerdo con los especialistas en el negocio. Este acuerdo no es un análisis funcional, sino una descripción de las necesidades de negocio.

2. **Modelado**. Los modelos formales son la transformación de esa comprensión en las ontologías estándar de ScooP. Las principales ontologías son:
    - Semántica del negocio: que contiene conceptos, atributos, relaciones, taxonomías, y otras definiciones. En este apartado se proporciona un soporte a plantillas que proveen clases con funcionalidades por defecto.
    - Interfaces de usuario: que van a describir los diferentes aspectos de la interrelación de los seres humanos con los objetos de negocio. Esto incluye la navegación sobre el sistema, los roles y los aspectos de seguridad, formularios y cuadros de diálogo, paneles y cuadros de mando.
    - Business Process Management (BPM): representa los flujos de trabajo formales para el problema de la empresa. Los flujos de trabajo están asociados a objetos de negocio, haciendo uso de metadatos y funciones, y también interfaces de usuario propias.

Los modelos se crean en dos capas de complejidad: modelado funcional, que se realiza por consultores y modeladores funcionales, que trabajan junto con los usuarios finales/expertos de dominio; modelado técnico, que cubre los aspectos técnicos de un sistema, tales como la integración con otros sistemas, algoritmos, optimización, reglas de negocio técnicas, etc. Modeladores con trasfondo técnico se encargan de esta parte. Tanto los modelos funcionales como los técnicos se mantienen sincronizados con ScooP, usando transformaciones modelo a modelo.

3. **Producción**. Generación de la aplicación de negocio. El sistema es generado como una serie de ficheros que contienen las especificaciones del sistema y sus metadatos, en forma de lenguajes específicos de dominio (DSLs). El paquete generado se instancia en un entorno para que se pueda hacer uso del mismo. En las etapas tempranas del proyecto normalmente se hace un pase a un entorno pre-producción de pruebas que no es definitivo. Posteriormente se instalará en un entorno de producción.

4. **Verificación**. La validación es la comparación de la solución que se ha generado con las necesidades y expectativas del negocio. Aquí, comprobamos que la transferencia de conocimiento entre los usuarios y los modeladores ha sido bueno, y que la representación práctica de las ideas es viable. La validación consta de dos aspectos:
    - Ajuste técnico: los componentes técnicos son validados, tales como las interfaces, las reglas de negocio, etc.
    - Ajuste de negocio: los especialistas del negocio revisan el sistema realizando casos de uso, representando sus situaciones reales.

Las primeras iteraciones en la espiral siempre producen una gran cantidad de conocimiento e información, mientras los expertos del negocio tienen oportunidad de expresar su opinión sobre el sistema; generalmente se les presenta un prototipo de la solución que pueden probar y con el que pueden trabajar para tomar decisiones que afectarán al desarrollo posterior. Esto hace que se refine y detalle mejor el problema, proporcionando cada vez una mejor comprensión. Cada iteración produce una visión mejor de la solución final, hasta que por fin se despliega el producto.

Debido a que el desarrollo está basado en modelos, a partir de la recogida de los requisitos del cliente se realiza un diseño inicial de la solución, que incluye la implementación de un prototipo funcional. Esto significa que ya en las fases de diseño en las que se comienzan a definir modelos prototipo, en realidad ya se está empezando a implementar parte del sistema final. Generalmente a partir de los modelos conceptuales y de interfaces, podemos obtener los modelos finales de datos y un modelo de proceso (*workflow*) bastante cercano a la realidad, que dejaría el sistema sólo con la parte técnica pendiente.

El siguiente diagrama ilustra la forma de trabajo y de desarrollo seguido para implementar la solución utilizando esta metodología, si bien se clasifican las distintas fases siguiendo un ciclo de vida más tradicional. En las siguientes secciones se explica de forma detallada cada una de ellas.

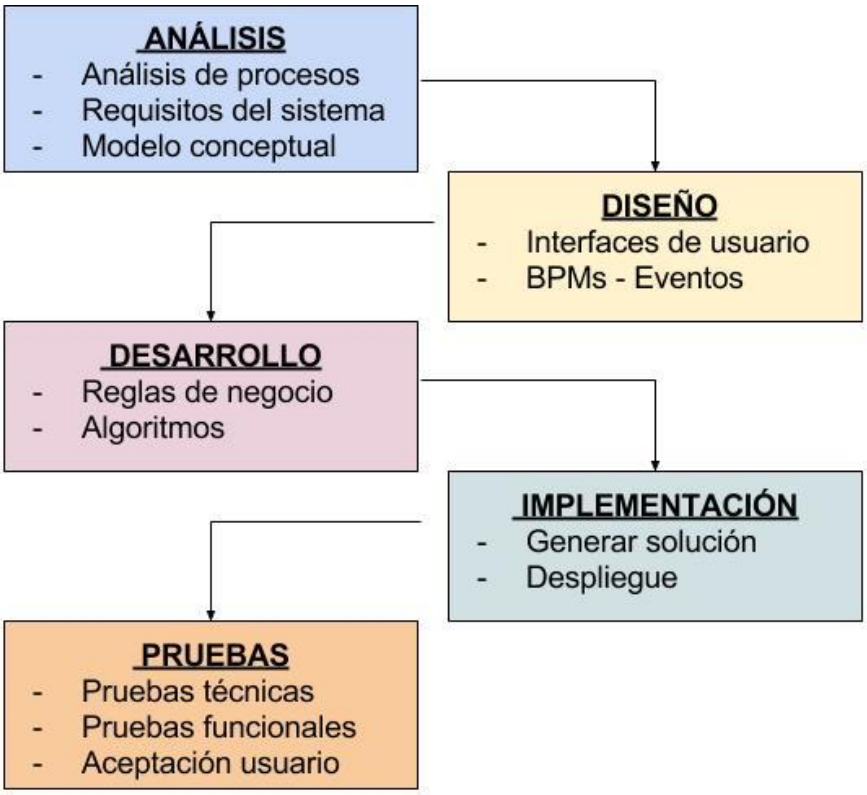

Figura 3: Ciclo de desarrollo

4 Análisis

En este apartado se tratará el análisis de los requisitos extraídos para el sistema. Está dividido en dos secciones: en la primera, se expondrá una visión global del sistema, con una descripción general de los requisitos agrupados; la segunda sección contendrá un detalle de los requisitos propiamente dichos.

4.1 Requisitos generales del sistema

Nos encontramos ante un sistema de ticketing a medida cuyos requerimientos varían dependiendo del tipo que se está tratando, pero que presentan una serie de características comunes, con lo que el requisito se especifica una vez, pero aplica a los diferentes módulos que hemos desarrollado. Hemos agrupado los requisitos según el área al que afectan, de la siguiente manera:

- **Alta/registro en el sistema**: son todos aquellos requisitos que tienen que ver con la entrada de datos en cada módulo. Por norma general, habrá un formulario de alta para cada ticket, con los campos especificados, y se impondrán unas reglas de validación que incluyen datos que son obligatorios y restricciones en la selección de valores, que aseguran que la información introducida por el usuario sea lo más útil, ordenada y real posible. Un ejemplo de este tipo de requisitos sería una validación de fechas, ya que tiene que haber un mínimo de días desde que se registra el ticket hasta que se pueda gestionar.
- **Consulta de la información**: toda la información que ha introducido un usuario en el sistema debe ser fácilmente accesible y además debe permitir consultar los detalles de los elementos relacionados con el ticket. Esto significa que, más allá de una simple visualización de los datos, hay requerimientos en cuanto a acciones que se pueden tomar desde la consulta del ticket (añadir un comentario, cambiar la prioridad), un cómodo acceso a la información que se maneja como usuario, y la posibilidad de realizar búsquedas y filtros sobre el conjunto total de los tickets del sistema.
- **Requisitos del flujo:** aunque la definición de los flujos y su implementación se especifican en el siguiente apartado, aquí se recogen todos los requisitos funcionales que tendrán que ser desarrollados junto con el BPM. Aquí se incluyen las interfaces necesarias en ciertos momentos del proceso o funciones que alteren el flujo normal, como puede ser una cancelación del proceso, o un reinicio por modificación de datos, por ejemplo.
- **Journal:** todos los tickets presentan un registro/log de todas las acciones efectuadas sobre el mismo, a modo de "diario" y para que todos los usuarios involucrados puedan llevar el seguimiento de lo que ha ocurrido o lo que falta por resolver. Este log es muy importante ya que también sirve como medio de comunicación entre el usuario solicitante y los técnicos encargados del ticket.
- **Gestión de parámetros:** reservado a un ámbito de administración del sistema, aquí se agrupan los requisitos que tienen que ver con los elementos de un ticket que son administrables, es decir, aquellos parámetros de configuración que son accesibles en el ticket pero que pueden variar a lo largo del tiempo. Para cada tipo de ticket hay una sección de administración que permite la configuración interna de ese ticket.
- **Emails:** otro requerimiento importante del sistema es el envío de emails automáticos. Este tipo de requisitos tiene que ver con qué información debería ser

enviada en los correos que lanza el sistema, así como alguna funcionalidad específica que requiera un desarrollo aplicado a esta comunicación.

4.2 Requisitos específicos

El listado siguiente ilustra los requisitos especificados por el cliente y que se han desarrollado como parte de este Trabajo de Fin de Grado. Cada requisito está identificado con un código que utilizaremos en otras secciones para referirnos a él. Estos códigos están formados con un prefijo que indica el tipo de ticket (INC – incidencias, PAR – parámetros, CHG – cambios, REQ - peticiones), el grupo de entre los que se han definido previamente (ALTA – registro de datos, CONS – consulta, BPM – requisitos de flujo, JOUR - journal, ADM – administración, EM – emails, GEN - general), y un número de secuencia. También se expone una breve descripción del requisito. Con el objetivo de simplificar la catalogación de requisitos, muchos de ellos agrupan varias funcionalidades, facilitando su comprensión y otorgando mejor estructuración.

En este apartado no se tienen en cuenta como requisitos los datos que se introducen en los formularios de registro de cada tipo, ya que se hablará de ellos en la sección de modelos de interfaz. Aquí se hacen referencia a los requisitos funcionales que requieren de un desarrollo técnico.

General

- **[GEN-001] Asignación de permisos a los grupos resolutores.** Los usuarios del sistema se pueden agrupar en grupos de trabajo que serán seleccionados para asesorar los distintos tipos de ticket. Es necesario que se pueda definir de una forma sencilla sobre qué tipo actúa cada grupo (puede ser uno o varios). Para ello, se solicita una interfaz específica de administración de los grupos con un selector de tipos. También se solicita para los usuarios poder asignarles un rol específico para el ticket de cambios, siendo éste aprobador de cambios o coordinador.
- **[GEN-002] Agrupación de tareas BPM por tipo de ticket.** El módulo de BPM trae consigo un punto de acceso de tareas, en el que se puede acceder de forma directa a la toma de decisiones. Aunque resulta muy útil debido a que muestra en todo momento las tareas que tiene pendientes el usuario conectado, al disponer de tantos procesos en los diferentes tickets queda confuso y desorganizado. Se solicita dividir las tareas pendientes según los tipos de ticket.
- **[GEN-003] Especificaciones Journal.** Como se ha mencionado anteriormente, el seguimiento de cada ticket en el sistema requiere de un registro de actividad sobre el mismo. Para ello, se solicita un log que combine tanto las acciones que se pueden realizar sobre el ticket (siendo estas la creación, consulta o modificación), como la toma de decisiones sobre el flujo (es decir, qué ocurre cuando el solicitante no tiene control sobre su ticket o qué decisiones han tomado los técnicos), además de la posibilidad de añadir comentarios de texto libre como entradas del log.

Ticket de incidencias

- **[INC-CONS-001] Acciones en consulta: cambiar prioridad, tomar decisión, añadir comentarios.** Una vez finalizado el registro en el sistema, no es posible modificar los datos introducidos por el usuario. Sin embargo, se necesita por parte de un técnico que trabaje con el ticket que sea posible re-catalogar la incidencia o reasignar ciertos valores del ticket que considere mal rellenados. Para ello se

requiere de una serie de acciones que se realicen sobre la consulta de la incidencia. Estas acciones permitirán modificar el campo de prioridad, tomar decisiones directamente desde el visor de datos (evitando tener que acceder a la pestaña propia del proceso), y añadir comentarios al journal. Esta última acción será accesible tanto por los técnicos del grupo que tiene asignada la incidencia, como el solicitante (posibilitando así un medio de comunicación entre ambos); el resto de acciones está limitado a los técnicos.

- **[INC-CONS-002] Enlace masivo de incidencias.** El ticket de incidencias presenta en su flujo una opción para identificar que la incidencia es repetitiva, lo que permite ligar la incidencia con otra y dejarla pendiente de la resolución de la incidencia padre. Cuando por ejemplo se registra en la empresa una caída del servidor de correo, se registrarán en el sistema muchas incidencias de la misma índole. Para poder trabajar con un único ticket y resolver varios de forma masiva, se pide una nueva funcionalidad de 'link' masivo de incidencias. Esta acción añadirá una nueva pestaña a la consulta de la incidencia donde se podrán visualizar todas las incidencias ligadas a esta. También es necesario poder tener una interfaz que permita filtrar y asociar incidencias de forma sencilla.
- **[INC-CONS-003] Desvincular una incidencia.** Relacionado con el tema anterior, es posible que una incidencia se ligue a otra sin tratar el mismo asunto, o simplemente por error. Se solicita implementar una función que desvincule una incidencia hija de su padre, marcando su estado en progreso de nuevo.
- **[INC-CONS-004] Link de consulta al solicitante y al grupo resolutor.** Se pide implementar una consulta directa desde la incidencia a la información propia del solicitante (con todos sus datos de usuario) y la información del grupo resolutor (qué miembros constituyen el grupo).
- **[INC-CONS-005] Buscador de incidencias.** Poder buscar y filtrar en el conjunto total del sistema por todos los campos importantes de una incidencia. Si no se encuentran resultados, mostrar un mensaje indicándolo y permitiendo volver a realizar la búsqueda (esto es, que no se produzca navegación si no hay resultados de la búsqueda).
- **[INC-CONS-006] Indicador de ficheros adjuntos y nuevo comentario en las listas de incidencias.** Con el objetivo de facilitar la mayor información posible de un solo vistazo, en el punto de acceso con el listado de las incidencias se solicita añadir dos columnas al inicio, con indicadores de si el ticket contiene ficheros adjuntos, y si se ha realizado un comentario por parte del solicitante mientras la incidencia está siendo tratada por los técnicos. De esta forma, los técnicos al trabajar con la herramienta consultarán la cola de trabajo con mejor criterio.
- **[INC-BPM-001] Lanzar el proceso inmediatamente después del alta (síncrono).** Este requerimiento surge de la necesidad de que el proceso se lance inmediatamente después del alta en el sistema de la incidencia. Inicialmente, el lanzamiento de los procesos se ejecutaba por medio de un disparador (*trigger*) para evitar la carga del proceso y mejorar el rendimiento general del sistema. Si bien el registro de tickets era más eficiente, un técnico que abriera un ticket no podría comenzar a trabajar en él de manera inmediata, debía esperar a que el disparador lanzase todos los procesos encolados. Como solución se propuso lanzar de forma síncrona el proceso al terminar el alta, con el contrapunto del rendimiento en el registro (la carga del proceso genera un pequeño retardo tras insertar la información).
- **[INC-BPM-002] Asignar a *helpdesk* cuando la prioridad es Alta/Crítica.** El *helpdesk* es un grupo de trabajo especial que proporciona soporte técnico de forma

constante (24x7). Además, son los encargados de manejar las incidencias que llegan con un nivel de criticidad alto o muy alto. Cuando una incidencia se abre con prioridad media o baja, pero posteriormente se incrementa (re-catalogada por un técnico, porque el incidente conlleva mayor riesgo), es necesario que la incidencia sea reasignada a este grupo de forma automática. Esto implica que el proceso deja de estar asignado al grupo resolutor inicial, para pasar a la fase de catalogación por *helpdesk*, que realizará las decisiones necesarias para resolver el ticket.

- **[INC-EM-001] Poder añadir comentarios desde el email.** El sistema de ITSM genera una serie de emails automáticos cuando un usuario tiene que realizar alguna tarea del proceso BPM. Estos emails llevan asociada toda la información relevante del ticket, y presentan un pequeño control para toma de decisiones desde el email (sin necesidad de acceder a la herramienta). Dado que los solicitantes generalmente son usuarios que no accederán a la herramienta más allá del registro de incidencias, es necesario que cuando un técnico les solicite información puedan tomar la decisión desde su correo y además aportar la información necesario en forma de comentario/texto libre. Para ello, se pide poder añadir comentarios (del tipo del journal) desde el email enviado automáticamente.

Ticket de cambio de parámetros

- **[PAR-ALTA-001] La fecha de SLA (acuerdo de nivel de servicio) tiene que contemplar días laborables.** Durante el registro del cambio de parámetros es posible definir una fecha de implementación. Se puede seleccionar entre un SLA por defecto y una fecha específica (normalmente para cambios de urgencia que se realicen en menos tiempo). Es necesario que el SLA por defecto sea de 5 días laborables.
- **[PAR-ALTA-002] Si la fecha de SLA está definida por el usuario, la justificación es obligatoria.** Relacionado con el anterior, en caso de que la opción escogida no sea la de por defecto, la fecha introducida tiene que ser justificada, ya que de otra forma el cambio puede ser desestimado en su fase de análisis.
- **[PAR-CONS-001] Visibilidad bandeja y permiso de alta.** El módulo de cambio de parámetros en el sistema debe ser accesible sólo por aquellos grupos con permiso de cambio de parámetros ([GEN-001]). Además, dentro de esta limitación sólo un grupo concreto de gente puede dar de alta cambios de parámetros, para mantener un control de las peticiones que se realizan. Se pide añadir un elemento de administración que permita establecer uno de los grupos del directorio activo de Windows como creador de cambio de parámetros. Esta opción estará disponible desde la administración de ITSM.
- **[PAR-JOUR-001] Entrada en el log si se produce rechazo por usuario aprobador.** El flujo de aprobación del usuario en un cambio de parámetros es independiente del proceso del propio ticket. Es por ello que no se añaden registros al journal cuando estos usuarios toman decisiones en sus procesos. Es necesario que cuando un usuario aprobador rechazar (con una razón justificada) se añada una entrada de log informativa, para que el solicitante y los técnicos tengan constancia.
- **[PAR-BPM-001] Asociar roles de maker-checker durante el proceso.** En el flujo de cambio de parámetros, después de realizar el análisis de viabilidad del cambio (el flujo completo se detalla en la sección 5), hay una serie de tareas que tienen que implementar los técnicos del grupo asociado al ticket. En concreto, hay que poder diferenciar entre el técnico que realiza la implementación del cambio de

parámetros (maker), y el que revisa y valida que ese cambio se haya realizado correctamente (checker). Estos técnicos no pueden ser el mismo dentro del grupo, por lo que tiene que haber un control del técnico que toma decisiones como realizador o como revisor, y validar que no sea el mismo, indicándolo en un mensaje de producirse este caso.

- **[PAR-BPM-002] Poder definir usuarios aprobadores.** En la fase de análisis de viabilidad, es posible que el analista determine que el cambio que le solicitan requiere de aprobaciones adicionales. Para ello, se pide implementar la funcionalidad necesaria para que el analista pueda seleccionar usuarios del directorio activo de Windows y establecerlos como usuarios aprobadores del cambio. Estos usuarios tienen que pertenecer al sistema, y tendrán un proceso de aprobación independiente al flujo del ticket.
- **[PAR-BPM-003] El analista puede editar cualquier campo.** Similar a lo que ocurría con [INC-CONS-001], es necesario que el técnico analista pueda re-catalogar el cambio de parámetros. En este caso, la solución propuesta es que todos los campos relativos al cambio de parámetros (esto excluye la información del solicitante) puedan ser editados en la fase de análisis únicamente.
- **[PAR-BPM-004] Indicador de pruebas.** En la fase de análisis nuevamente, tiene que ser posible para el analista determinar si el cambio de parámetros requiere de una fase de *testing* antes de ser implementado de manera oficial, dando la posibilidad al solicitante de ver las consecuencias de su solicitud. Este indicador estará disponible en la fase de análisis y modifica el flujo normal de un ticket, añadiendo una pre-implementación del cambio y una validación por parte del solicitante.

Ticket de cambios

- **[CHG-ADM-001] Administración de tipologías de un cambio.** En el ticket de cambio, es posible definir una serie de categorías seleccionadas en unos controles de tipo *combo-box*. Estas categorías recogen los tipos de plan de restauración, las categorías y subcategorías de cambios, y los tipos de inactividad que pueden producir estos cambios. Estas listas obtienen sus valores de unas categorías definidas en el sistema, y que deben ser administrables. Esto significa que el administrador debe poder modificar, añadir o eliminar valores de esas listas.
- **[CHG-ADM-002] Plantillas.** De igual manera que se pueden definir valores de categorías en la administración, es necesario poder definir unas plantillas de texto para rellenar con una información inicial ciertos campos del ticket de cambios. Estos campos hacen referencia a los planes de pruebas, restauración y justificación del cambio, e incluyen una serie de preguntas que el solicitante debería responder. Estas plantillas deben ser modificables por el usuario administrador.
- **[CHG-ALTA-001] Poder subir múltiples ficheros.** Debido a la alta demanda de documentación que se solicita para realizar un cambio, es necesario disponer de un control de fichero múltiple en el registro del ticket, para añadir de golpe todos los documentos requeridos. También es necesario disponer de una pestaña en la consulta que permita visualizar todos los ficheros adjuntos.
- **[CHG-ALTA-002] La persona seleccionada como coordinador no puede ser el solicitante.** Validación en el registro de que el coordinador sea un usuario diferente del solicitante.
- **[CHG-ALTA-003] El periodo del cambio tiene que comenzar mínimo 5 días laborables a partir del alta.** Los tickets de cambio acarrean una serie de

aprobaciones y de implementación de tareas que los distinguen del resto de tickets disponibles. Por ello, sus restricciones de fechas son mayores. Es necesario que las fechas en las que se implemente el cambio estén controladas, y que la fecha de comienzo sea 5 días laborables a partir del registro del ticket en el sistema.

- **[CHG-CONS-001] Visibilidad bandeja y permiso de alta.** Igual que [PAR-CONS-001] pero para grupos con permiso de cambios.
- **[CHG-BPM-001] La fecha de expiración sólo afecta hasta la etapa de implementación.** Durante el registro del cambio, se establece una ventana de implementación. Este periodo incluye una fecha calculada automáticamente como un día antes del fin como fecha de expiración. Esta fecha indica el momento en el que el proceso del cambio debe cancelarse automáticamente por no llevarse a cabo a tiempo. Esta fecha sólo aplica a las fases de antes de la implementación, es decir, aquellas que implican la aprobación del cambio.
- **[CHG-BPM-002] Posibilidad de editar el cambio.** Durante el proceso del cambio se producen validaciones de grupos resolutores y tareas asociadas (el flujo se detalla en la sección 5). Es necesario que, si en algún momento se produce un rechazo, el solicitante pueda parar y editar la información que introdujo inicialmente. Esta edición permite al solicitante editar cualquier campo del ticket, pero reinicia por completo el proceso (debido a que se han producido modificaciones y es necesario que se valide nuevamente desde el principio).
- **[CHG-BPM-003] Poder definir tareas, grupos aprobadores y grupos revisores.** Siguiendo la filosofía de [PAR-BPM-002], durante el proceso del cambio el solicitante puede definir tareas que tendrán que ser aprobadas e implementadas por grupos resolutores, además de definir grupos específicos de aprobación adicional. También se ofrece la posibilidad de definir grupos revisores, que son grupos de usuarios que deberían ser avisados por correo electrónico de las iteraciones del cambio, pero que no forman parte activamente del mismo.
- **[CHG-BPM-004] Poder asociar servidores e instancias.** Proporcionar un mecanismo de documentación del cambio referente a los servidores e instancias de aplicaciones. Este apartado tiene que ver con la integración con la CMDB, y pretende dar la funcionalidad necesaria para que el solicitante pueda definir de forma sencilla qué servidores son afectados por el cambio, y que el sistema precargue de forma automática las instancias de aplicaciones presentes en esos servidores.
- **[CHG-JOUR-001] Entrada en el log si se produce rechazo por grupo aprobador o tarea.** Igual que [PAR-JOUR-001].

Ticket de peticiones

- **[REQ-ALTA-001] Incluir la información del solicitante.** Añadir en el formulario de alta la información relevante al solicitante. Esta información deberá estar precargada con la información del usuario conectado, pero puede ser modificada para realizar peticiones en nombre de otra persona.
- **[REQ-ALTA-002] Validación de plantillas.** Al dar de alta una petición el usuario tendrá la opción de descargarse la plantilla. Entonces tendrá que rellenarla (en su editor correspondiente) y subir el fichero relleno con la opción actual de Adjuntar ficheros. El sistema validará que el usuario ha subido un fichero si hay plantilla. Estas plantillas vienen definidas por la categorización de la petición en base a la aplicación seleccionada.

- **[REQ-ADM-001] Plantillas.** Poder definir la categorización mencionada en [REQ-ALTA-002] por parte de un administrador. Será administrable qué aplicación llevará plantillas asociadas, mostrando además en el formulario de registro de la petición los controles necesarios de ficheros de forma dinámica.
- **[REQ-BPM-001] Ficheros en la fase de transfer.** Durante el proceso de petición (especificado en la sección 5) es posible para un técnico transferir la petición a un grupo externo. En este contexto, deben poder seleccionarse ficheros a enviar en el email que se le envía a ese grupo desde el sistema. Se solicita disponer de una interfaz en el proceso que permita seleccionar entre los ficheros asociados previamente al ticket, y subir manualmente ficheros, además de poder escribir un texto adicional. Toda esta información será adjuntada en el email para el grupo.

5 Diseño

Dado que tratamos con un sistema basado en modelado de procesos, este apartado incluye toda la información y especificaciones de los distintos modelos. Se incluyen aquí todos aquellos elementos trabajados por el autor.

5.1 Casos de uso

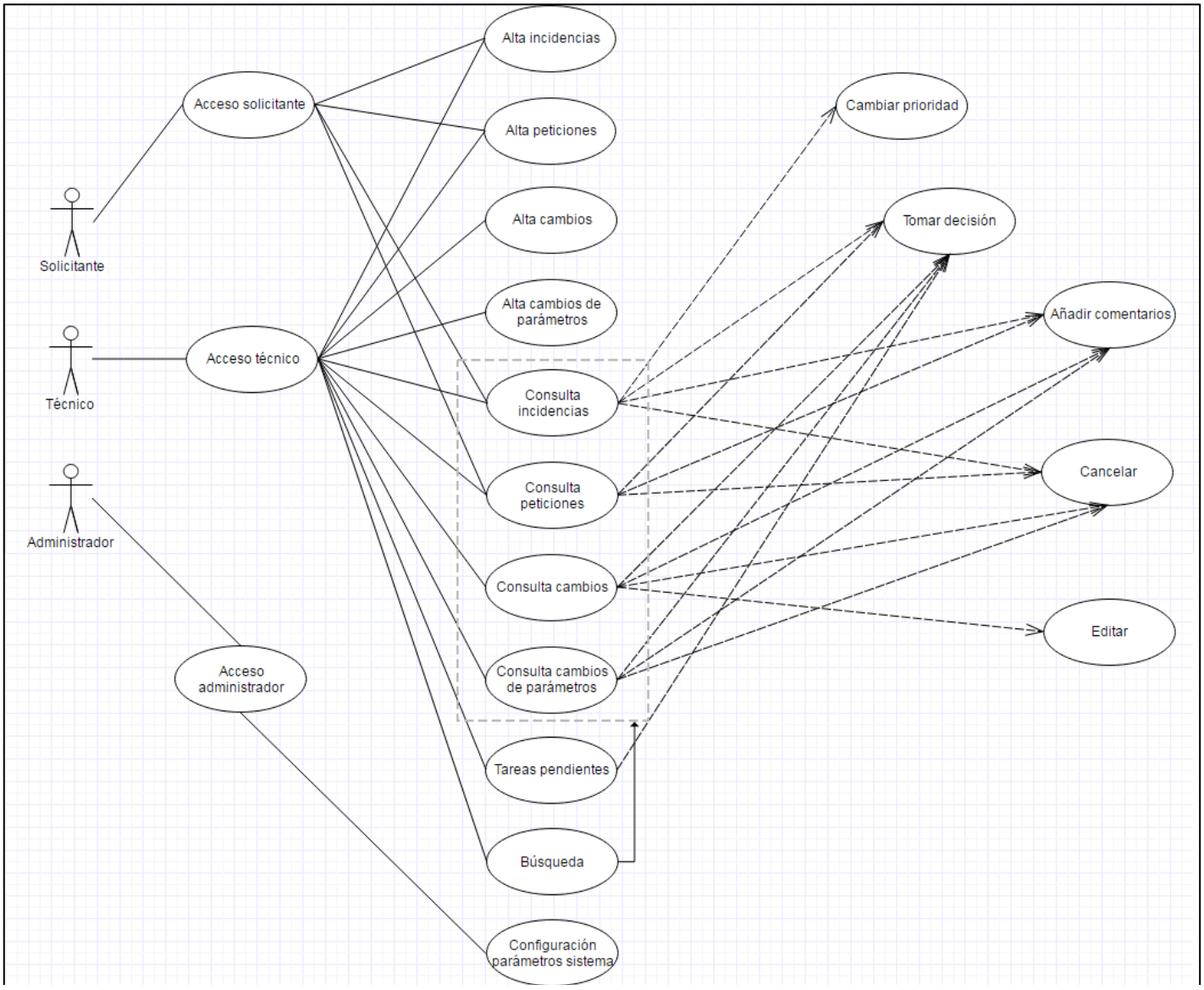

Figura 4: Casos de uso

Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema. Los casos de uso, representados en las burbujas, son una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso. Para relacionar todos los elementos disponemos de conectores de dos tipos: asociación (línea continua) que indica la invocación desde un actor o caso de uso a otra operación (caso de uso); y dependencia, en la que un caso depende de otro.

Como se puede apreciar en el diagrama, existen tres tipos de usuarios en el sistema. El usuario **solicitante** será capaz de dar de alta tanto incidencias como peticiones, pero no los otros dos tipos de ticket; además, será capaz de consultar el estado de sus solicitudes abiertas, y añadir comentarios en los mismos o cancelarlos (si ya se hubiera resuelto su problema, por ejemplo). Por su parte, el **técnico** accederá al sistema para realizar la gestión de todos los tickets, por lo que puede registrar cualquiera de los tipos, así como consultarlos y realizar acciones sobre ellos (añadir comentarios, tomar decisiones, cancelarlos, o modificar alguno de sus valores). Por último, el **administrador** dispone de un área especial dónde podrá acceder a la configuración de los parámetros de cada tipo de ticket.

5.2 Modelo de procesos

En este apartado se tratan todos los procesos de negocio representados en el sistema. El flujo propio de cada ticket, a excepción del proceso de Incidencias que no se incluye ya que no fue desarrollado para este Trabajo Fin de Grado.

BPM - Cambio de parámetros

El solicitante registra los datos del cambio de parámetros. La solicitud de cambio llega al responsable del departamento para realizar una aprobación; el técnico (maker) que aplique se hará cargo de la solicitud, analizará la viabilidad, revisará que la información es correcta, y establecerá el riesgo que la misma conlleva. En caso de que falte algo solicitará al solicitante la información adicional y la cumplimentará en la herramienta.

Cuando el técnico confirme la solicitud, esta llegará al responsable de gestión de parámetros para una nueva aprobación. Una vez el responsable de su visto bueno la solicitud pasa de nuevo al técnico (maker), que adjuntará en la herramienta la información de los parámetros previa al cambio y lo implementará. El cambio de parámetros pasará a otro técnico (chequer) encargado de revisar el cambio y adjuntar la información post-implementación. El cambio de parámetros pasará de nuevo al técnico (maker) para cerrar el cambio de parámetros. Se comunica al solicitante que el cambio ha sido ejecutado vía correo electrónico.

Después de las revisiones con el cliente, este proceso fue modificado: en la fase de registro se puede indicar si el cambio de parámetros requiere o no de un plan de pruebas que será realizado por el técnico (maker) antes de implementar el cambio; si las pruebas son satisfactorias para el solicitante, el cambio de parámetros prosigue de forma normal. También se añade una fase de aprobación opcional, esto es, el técnico que analiza la viabilidad del cambio de parámetros puede declarar usuarios que realicen aprobación

adicional; en caso de no declararlos, este paso se ignora. El flujo de cambio de parámetros se puede consultar en el Anexo A, a modo de ejemplo.

BPM – Cambios

El solicitante registra los datos del cambio. La solicitud llega al coordinador del cambio, que es un miembro del mismo grupo de trabajo que el solicitante y que debe dar una primera aprobación de los datos introducidos. Después, el solicitante recibe de vuelta el cambio para declarar todas las tareas a realizar durante el cambio, asociando a cada una un grupo encargado; también establecerá todos los grupos que deben aprobar la totalidad del cambio y sin los cuales el cambio no se podrá implementar. Por último, es posible definir una lista de grupos revisores que serán comunicados por email sobre los avances del cambio.

Una vez declaradas todas las tareas, se produce una fase de aprobación en la que todas las tareas tienen que ser aprobadas para poder proseguir con el cambio; si alguna de las tareas es rechazada, se produce un intercambio entre el solicitante y el grupo, a fin de llegar a un acuerdo y poder aceptar la tarea asignada. Cuando todas las tareas han sido aceptadas por sus grupos, el cambio queda pendiente de los grupos aprobadores, encabezados por el Comité de cambios; nuevamente, el flujo no avanza hasta que todos los grupos hayan dado su visto bueno y en caso de rechazo se produce un intercambio con el solicitante.
Tras la fase de aprobación, se implementa el cambio: todas las tareas declaradas serán realizadas por sus correspondientes grupos. Si algún grupo no consigue implementar su tarea, ésta quedará marcada como no implementada. Al final del proceso, el solicitante toma el control y procederá al cierre del cambio indicando si se trata de un cambio realizado correctamente, parcialmente (si alguna de las tareas no se ha podido realizar) o se ha implementado el plan de marcha atrás (han ocurrido errores durante el proceso y el cambio se cancela, volviendo al estado inicial de la aplicación).

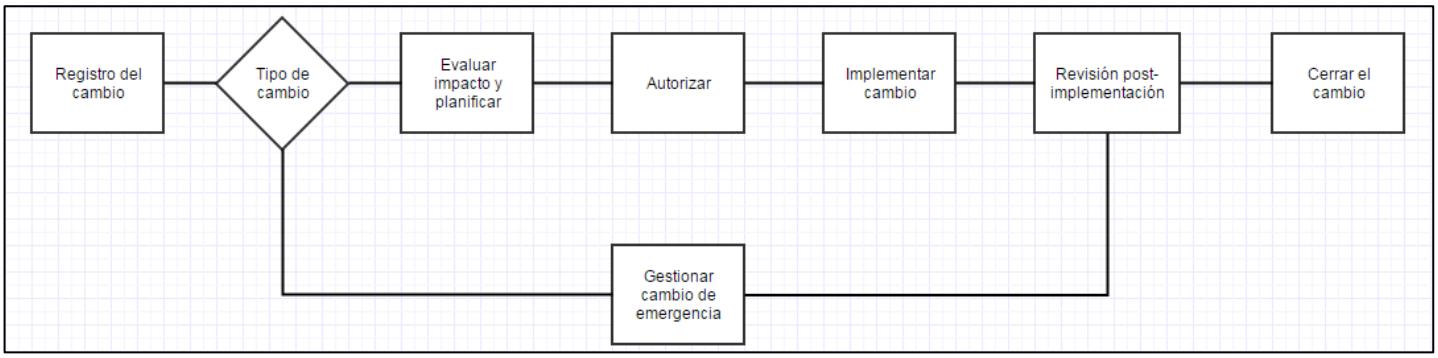

Figura 5: Etapas del flujo de cambios

BPM – Peticiones

Siguiendo el estilo de los procesos anteriores, el solicitante registra los datos de la petición. La petición se envía para aprobación del responsable del solicitante. En este momento el responsable puede decidir si es necesaria una aprobación adicional. Una vez aceptada, la petición pasa al grupo resolutor asignado para su resolución, que realizará un breve análisis para determinar si necesita más información por parte del solicitante, si realiza la petición, o remitirla a otro grupo para que lleve a cabo las acciones oportunas. Finalmente, el

solicitante retomará el control para determinar si su petición ha sido resuelta adecuadamente o, en caso contrario, reabrirla y comenzar de nuevo el proceso.

5.3 Modelo conceptual

En el sistema podemos encontrarnos con diferentes tipos de conceptos, diferenciando entre conceptos principales, conceptos intermedios (generalmente representados por una tabla que relaciona dos conceptos) o tipologías/taxonomías, que representan una lista cerrada de valores preestablecidos. Un concepto representa la totalidad de un objeto de negocio, con toda la información propia del objeto (datos), su lógica (en forma de funciones), los elementos gráficos que interactúan con dicha información (modelo de interfaces) y el modelo de estados. En este apartado se tratarán los conceptos desarrollados más característicos del sistema. Se diferenciará entre conceptos con persistencia (disponen de una tabla en base de datos) y conceptos sin persistencia.

Incidencia (ReqIncident)

Representa el ticket de incidencias. Se trata de un concepto principal con persistencia en base de datos que recoge toda la información del ticket y también toda la lógica de negocio detrás del mismo. Las incidencias tienen un BPM asociado (explicado en la sección de procesos) que provocará las transiciones entre los diferentes estados.

Estado	Descripción
OPEN	Abierta. Es el estado inicial, en el que se queda la incidencia después de haber sido registrada en el sistema.
PROCESS	En progreso. Cuando la incidencia comienza a ser tratada por los técnicos.
WAIT_USER_INFO	En espera de información. Un técnico requiere información adicional del solicitante y la incidencia se queda en espera de la misma.
ON_HOLD	En espera de otra incidencia. Si se asocia una incidencia a otra, la incidencia hija queda en estado de espera a que se resuelva su padre.
REOPENED	Reabierta. Una incidencia está reabierta cuando el solicitante no está de acuerdo con la solución que le han proporcionado los técnicos.
FINISHED	Finalizada. Una incidencia resuelta correctamente.
CANCELLED	Cancelada. Cuando una incidencia se ha cancelado por el solicitante o un técnico.

Tabla 1: Modelo de estados de incidencias

Cambio de parámetros (ReqChangeParams)

Es el concepto principal con persistencia que modela el ticket de cambio de parámetros y toda su funcionalidad asociada. Dispone de un BPM que establecerá las transiciones en el modelo de estados del concepto.

Estado	Descripción
OPEN	Abierta. Es el estado inicial, en el que se queda el cambio de parámetros después de haber sido registrado en el sistema.
ANALYSIS	Análisis. Primera fase del proceso, cuando el analista hace un estudio de la viabilidad.
APPROVAL	Aprobación. El cambio de parámetros se encuentra en esta fase cuando está pendiente de la aprobación por parte de un responsable.
EXECUTION	Ejecución. Una vez se ha aprobado, el proceso actualiza el ticket automáticamente a este estado, en el que se mantiene hasta que un técnico resuelve la solicitud.
CLOSURE	Cierre. El cambio de parámetros está pendiente de que se revise la implementación.
RESOLVED	Resuelto. El cambio ha sido resuelto de forma correcta.
RESOLVED_KO	No resuelto. El cambio ha sido resuelto de forma incorrecta, no satisface las necesidades solicitadas.

Tabla 2: Modelo de estados de cambio de parámetros

Solicitud de cambios (ReqChange)

Concepto principal con persistencia en base de datos que representa el ticket de cambios. Contiene toda la información del ticket y su lógica de negocio. Se caracteriza por tener asociado un BPM propio que genera objetos de tipo tarea y grupos aprobadores y revisores.

Estado	Descripción
OPEN	Abierta. Es el estado inicial, en el que se queda el cambio después de haber sido registrado en el sistema.
EDITION	Edición. Durante el proceso de cambios, el solicitante puede decidir editar la información previamente introducida. Este estado refleja la situación en la que el cambio está siendo editado, pero no se ha relanzado el proceso.
REVIEW	Revisión. Primera fase del proceso, cuando el analista hace un estudio de la viabilidad.
TASK_ASSESMENT	Evaluación de tareas. El cambio de se encuentra en esta fase cuando está pendiente de la aprobación de tareas.
GROUP_APPROVAL	Aprobación de grupos. El cambio de se

	encuentra en esta fase cuando está pendiente de la aprobación de grupos.
IMPLEMENTATION	Implementación. El cambio se encuentra en espera de que todas las tareas se lleven a cabo.
CANCELLED	Cancelado. El cambio ha sido cancelado.
CLOSURE	Cierre. Se revisa la implementación del cambio.
FINISHED	Finalizado. El cambio ha sido resuelto de forma correcta.

Tabla 3: Modelo de estados de cambios.

Petición (Request)

Concepto principal con persistencia que representa el ticket de peticiones. Nuevamente, dispone de un BPM que modifica el estado de la solicitud.

Estado	Descripción
PENDING_APPROVAL	Pendiente de aprobación. Es el estado inicial, en el que se queda la petición después de haber sido registrada en el sistema.
PENDING_ASSIGNMENT	Pendiente de asignación. Una vez aprobada por el primer responsable, la petición se queda esperando a que sea asignada a un grupo.
PROCESS	En proceso. La petición está siendo tratada por los técnicos.
WAIT_USER_INFO	En espera de información. Un técnico requiere información adicional del solicitante y la petición se queda en espera de la misma.
PENDING	Pendiente. Es un estado en el que se puede quedar la petición cuando un técnico deja la solicitud en espera de forma voluntaria.
REOPENED	Reabierta. Una incidencia está reabierta cuando el solicitante no está de acuerdo con la solución que le han proporcionado los técnicos.
CANCELLED	Cancelada. La petición ha sido cancelada.
REJECTED	Rechazada. La petición ha sido rechazada por cumplimentación errónea o por no aceptación del grupo técnico.
FINISHED	Finalizado. La petición ha sido resuelta de forma correcta.

Tabla 4: Modelo de estados de petición

Tareas del ticket de cambios (TaskITSM)

Concepto con persistencia en base de datos que representa las tareas que se crean durante el proceso de cambios. Debido a la información que se maneja y que tiene un BPM propio, se considera como un concepto completo. El modelo de estados es muy similar al de la solicitud de cambios, de la que depende para avanzar. No se incluye su tabla por motivos de síntesis de la memoria. Sus estados serían los siguientes:

OPEN – PENDING_ACCEPTANCE – PENDING_IMPLEMENTATION – IMPLEMENTATION – REJECTED – CANCELLED – RESOLVED – RESOLVED_KO

Grupos aprobadores/revisores (RelGrupoChange)

Se trata de conceptos intermedios que representan una relación N-N entre dos conceptos principales. En este caso, en la fase de análisis del ticket de cambio se definen grupos de trabajo que tendrán que tomar decisiones sobre el cambio. Estos grupos son los grupos aprobadores, y se representan mediante este concepto intermedio. Disponen de un pequeño BPM de aprobación (aceptar o rechazar simplemente). Es un concepto de relación con persistencia en base de datos.

Servidores e instancias (RelReqChgInstance/RelReqChgServer)

Concepto intermedio que relaciona los servidores y las instancias (ambos son objetos extraídos de la CMDB) con el ticket de cambios. Durante la fase de análisis y creación de tareas es posible asociar a la solicitud los servidores e instancias que serán afectados por el cambio. Se trata de un concepto de relación con persistencia en base de datos pero que no dispone de un proceso propio.

Tipologías o listas administrables

Durante el desarrollo del proyecto se han establecido una serie de listas que han sido precargadas en el sistema y que aportan información seleccionable durante el registro de los tickets, esto es, que en el formulario de alta dispondremos generalmente de controles tipo combo con unos valores predeterminados en el sistema. Ciertos valores deben poder ser editados por un administrador, en caso de que se quiera cambiar el nombre o incluir alguno nuevo, así como eliminar otros. Para ellos contamos con una serie de conceptos denominados tipologías que se almacenan en base de datos y que cuentan básicamente con un código y una descripción. Estos conceptos no disponen de modelo de estados, y en su lugar se indica si están activos o inactivos mediante un flag. En este proyecto se han tenido que definir las siguientes tipologías: tipos de plan de restauración (BackOutPlanType), categorías de cambio (ChangeCategory), subcategorías de cambio (ChangeSubCategory), categorías de inactividad (DowntimeCategory).

5.4 Modelo de datos

En este apartado se mostrará de forma detallada el modelo de datos desarrollado para el sistema. Se mostrarán un ejemplo de las tablas más significativas, indicando en primer lugar el nombre del atributo, seguido del tipo, y finalmente una breve descripción de lo que representa. Los tipos de datos que se manejan son: numérico (generalmente entero, en caso de ser decimal se especificaría como dec.), fecha, texto (de longitud variable, normalmente a partir de 100 caracteres), código (de longitud variable, normalmente máximo 20 caracteres) y flag (1 único caracter).

Indica que se trata de una clave externa. Indica que es una clave primaria.

5.4.1 ITS_CHANGE_PARAM

Tabla de cambio de parámetros. Se muestra en detalle la información que almacena el ticket de cambio de parámetros. Sirve como ejemplo del resto de tipo de ticket del sistema, ya que, aunque pueden variar en los datos que se almacenan, siguen una estructura similar. Se han tenido que definir en total 5 tablas de este estilo, una para cada tipo de ticket, y una para el concepto de tareas (TaskITSM).

Atributo	Tipo de dato	Descripción
REQCHANGEPARAM_ID	Numérico	Identificador de la tabla. El valor de un identificador se extrae de un secuencial que se genera cada vez que se crea una instancia del concepto.
ALTA_DT	Fecha	Fecha de alta en el sistema.
ALTA_USR	Numérico	Usuario de alta. Relacionado con la tabla de usuarios del sistema
APPCATEGORY_CD	Numérico	Categoría de aplicación. Es un atributo relacionado con las categorías de aplicaciones definidas en la CMDB. Se utiliza para filtrar las aplicaciones que afectan al ticket.
APPLICATION_ID	Numérico	Identificador de la aplicación. Está relacionado con la aplicación de CMDB.
BUSINESS_OTHER_DS	Texto	Atributo de texto libre introducido por el usuario cuando no encuentra el tipo de negocio correspondiente.
BUSINESSTYPE_DS	Texto	Indica el tipo de negocio al que afecta el cambio de parámetros. Es un valor que se escoge de una lista cerrada (fraude, legal, ventas…).
CHECKER_USR	Numérico	Usuario técnico que toma las funciones de revisor durante el proceso del cambio de parámetros. Relacionado con la tabla de usuarios del sistema.
ESTADO_CD	Código	Indica el estado en el que se encuentra el concepto.
ESTADO_DT	Fecha	Fecha en la que se produjo el salto al

		estado actual.
DATE_EXPECTED_CD	Código	Atributo que indica si el cambio de parámetros se va a hacer en una fecha con SLA estándar o en una fecha determinada por el usuario.
EXPECTED_DT	Fecha	Fecha esperada de realización del cambio.
JUSTIF_REQ_DS	Texto	Justificación del cambio.
MAKER_USR	Numérico	Usuario técnico que implementa el cambio de parámetros durante el proceso. Relacionado con la tabla de usuarios del sistema.
MOD_DT	Fecha	Fecha de modificación del concepto.
MOD_USR	Numérico	Usuario de modificación.
MODULO_APP_ID	Numérico	Identificador del módulo de la aplicación. Las aplicaciones pueden disponer de una categorización adicional denominada módulo. Proviene de la CMDB.
PRUEBAS_FL	Flag	Bandera que indica si el ticket requiere de pruebas pre-instalación.
REQCHANGEPARAM_CD	Código	Código del ticket. Sigue el formato PAR_AÑO_SECUENCIA
REQUEST_USR	Numérico	Usuario solicitante del cambio de parámetros. Generalmente es el usuario conectado.
RESUMEN_DS	Texto	Texto resumen del cambio de parámetros. También se utiliza como texto libre para notas del solicitante.
RISK_CD	Código	Indica el tipo de riesgo que supone realizar el cambio, puede ser Bajo, Medio, Alto.
TITULO_DS	Texto	Título del cambio de parámetros.

Tabla 5: Tabla cambio de parámetros

5.4.2 ITS_RELGRUPOCHANGE

Tabla de grupos relacionados con el cambio. Tabla intermedia que representa la relación N-N entre el concepto ReqChange y GrupoTrabajo. Se han tenido que definir en el sistema 2 tablas de este tipo, una para el ticket de cambios y otra para el de parámetros.

Atributo	Tipo de dato	Descripción
RELGRUPOCHANGE_ID	Numérico	Identificador de la tabla. El valor de un identificador se extrae de un secuencial que se genera cada vez que se crea una instancia del concepto.
ALTA_DT	Fecha	Fecha de alta en el sistema.
ALTA_USR	Numérico	Usuario de alta. Relacionado con la tabla de usuarios del sistema

GROUP_TYPE_CD	Código	Tipo de grupo. Los grupos relacionados con el cambio pueden ser aprobadores o revisores.
GRUPO_TRABAJO_ID	Numérico	Identificador del grupo de trabajo. Relacionado con la tabla de grupos de trabajo del sistema (GrupoTrabajo).
REQCHANGE_ID	Numérico	Identificador del cambio (ReqChange).

Tabla 6: Tabla de relación cambio – grupo trabajo

5.4.3 ITS_RELREQCHGSERVER

Tabla de servidores relacionados con el cambio. Tabla intermedia que representa la relación N-N entre el concepto ReqChange y Server. Se han tenido que definir en el sistema 2 tablas de este tipo, una para servidores y otra para instancias (Instance).

Atributo	Tipo de dato	Descripción
RELREQCHGSERVER_ID	Numérico	Identificador de la tabla. El valor de un identificador se extrae de un secuencial que se genera cada vez que se crea una instancia del concepto.
COMPONENTE_ID	Numérico	Identificador del componente, en este caso servidor.
REQCHANGE_ID	Numérico	Identificador del cambio (ReqChange).

Tabla 7: Tabla de relación cambio - servidor

5.4.4 RS_GRUPO_TRABAJO

Tabla que representa los grupos de trabajo del sistema. Contiene la información específica del grupo, los permisos del que dispone, y la información referente al directorio activo (los grupos de trabajo del sistema están mapeados contra el directorio activo de Windows del cliente).

Atributo	Tipo de dato	Descripción
GRUPO_TRABAJO_ID	Numérico	Identificador de la tabla. El valor de un identificador se extrae de un secuencial que se genera cada vez que se crea una instancia del concepto.
ACTIVE_DIRECTORY_DS	Texto	Nombre del grupo de trabajo en el directorio activo.
ACTIVE_DIRECTORY_FL	Flag	Bandera que indica si el grupo de trabajo se sincroniza con el directorio activo.
BORRADO_FL	Flag	Bandera de borrado lógico.
GRUPO_TRABAJO_CD	Código	Código del grupo para identificarlo en el sistema.
GRUPO_TRABAJO_DS	Texto	Nombre del grupo de trabajo

Tabla 8: Tabla de grupo de trabajo

5.5 Modelo de interfaces

En este apartado vamos a exponer el diseño habitual de una interfaz del sistema. presentaremos los elementos que suelen ser comunes a todas las pantallas, así como los elementos del modelo de interfaz con los que podemos trabajar.

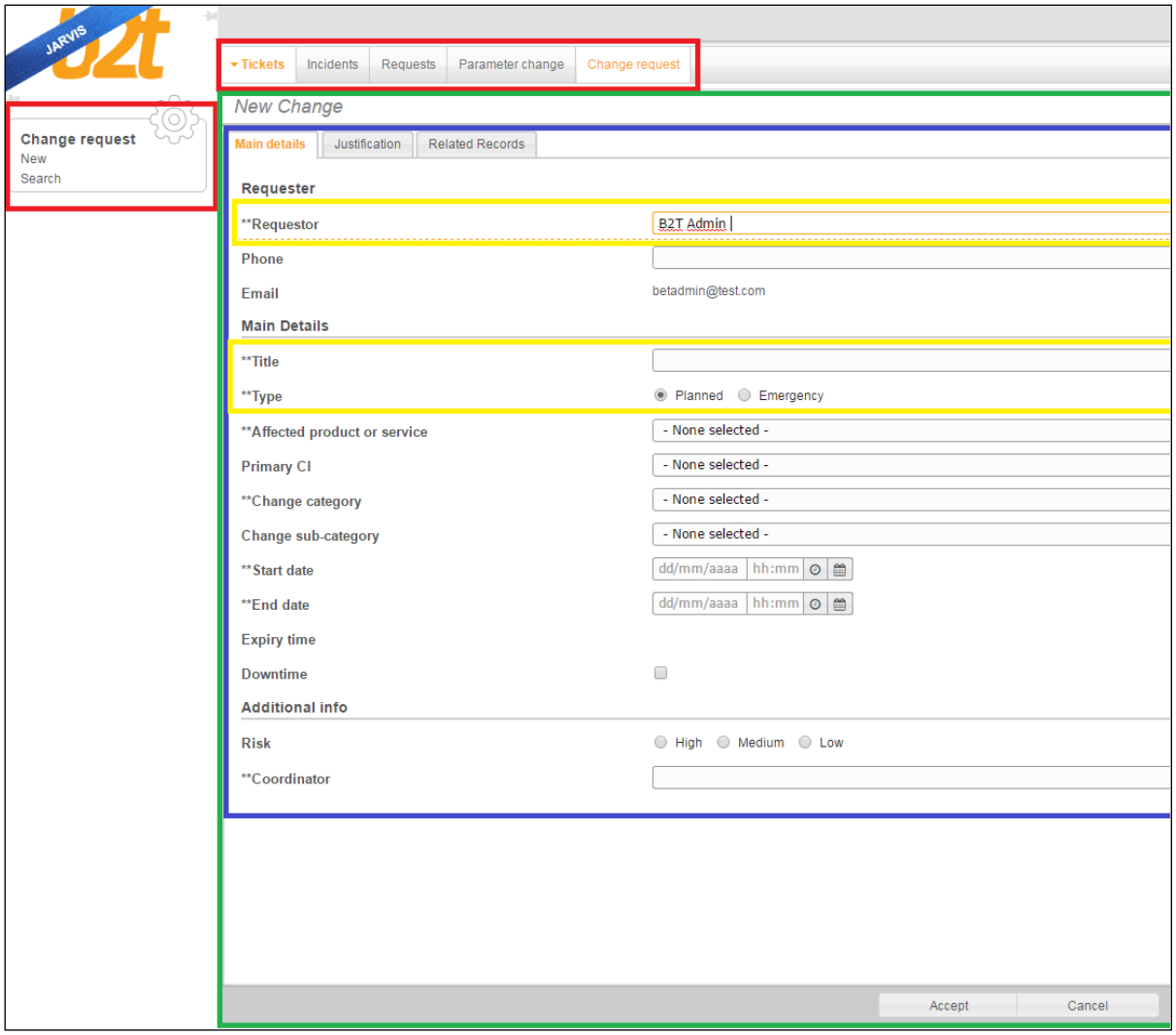

Figura 6 - Estructura de una interfaz

La figura 6 presenta el modelo típico de interfaz. Cuando el sistema se entrega al cliente se cambian algunos aspectos de la misma como los colores, y los logos.

La estructura de la interfaz es la siguiente:

- En la parte superior y lateral izquierdo (resaltados en rojo) disponemos de barras de navegación. La barra de navegación superior está conformada por bandejas, que son agrupadores de elementos de navegación; una bandeja puede estar formada por uno o más puntos de entrada al sistema llamados puntos de acceso. Al pinchar sobre un punto de acceso se producirá una navegación en el sistema, normalmente para acceder a una primera visión en forma de lista. En el lateral izquierdo

disponemos de las acciones que se pueden realizar, generalmente la creación de nuevos elementos o el acceso a un buscador para poder filtrar las listas.

Salvo excepción, cada concepto principal del sistema dispone de una clase punto de acceso donde se configuran estos elementos de navegación y pantallas específicas de entrada. Estas barras de navegación se pueden configurar con los elementos deseados y limitar su acceso dependiendo de los usuarios mediante el modelo de seguridad, que determina qué puede ver cada tipo de usuario.

- En la parte central se presenta el elemento página (resaltado en verde). Una página es un contenedor de elementos, un "marco" para otros elementos. Todas las páginas se pueden configurar con hasta tres botones que ejecuten funcionalidad o una navegación. Los botones más comunes son 'Aceptar', 'Cancelar' y 'Guardar Cambios', aunque son totalmente configurables y pueden realizar cualquier función.

- En el interior de la página se encuentran los visores (resaltado en azul). Una página puede tener uno o más visores (en cuyo caso aparecen en forma de pestañas). Estos visores están formados por elementos individuales denominados controles (como ejemplos, los encuadrados en amarillo). Los controles pueden ser de diversos tipos: texto, numérico, lista, botón de búsqueda, radio button, multicheck, combo, toolbar. Estos controles se utilizan para la gestión de la información en todo momento. Permiten definir formularios de alta, de búsqueda, gestión de listas, etc. y almacenan o extraen la información de la base de datos.

5.6 Modelo de seguridad

El modelo de seguridad define qué pueden ver y qué pueden hacer los usuarios que acceden al sistema, mediante una estructuración de bandejas, agrupadas en grupos de seguridad, que se añaden a los diferentes roles presentes.

5.6.1 Headers o Bandejas

Son los puntos de entrada a la aplicación. Cada bandeja accede a una visión inicial definida, que en general suele ser un listado mostrando el concepto definido del punto de acceso. Por ejemplo, habría una bandeja de acceso a las incidencias, que mostraría un listado de las incidencias del sistema (este listado puede estar filtrado para que vea sólo lo que me interesa como usuario conectado). También se pueden hacer agrupaciones de bandejas, de forma que tenga una bandeja que me permita acceder a varios puntos de acceso del mismo tipo. En el sistema ITSM se aprecian las siguientes agrupaciones: ‘Mis peticiones’, que representa el punto de entrada del usuario solicitante (véase diagrama de casos de uso), y que le permite acceder a un panel principal con un listado de sus incidencias y sus peticiones; ‘Peticiones’, que es el punto de entrada del técnico, consiste en una agrupación de los distintos tickets, pudiendo así acceder a cada tipo de ticket filtrado por lo que debe trabajar; por último disponemos de ‘Administración ITSM’, donde se encuentran todos los puntos de acceso referentes a la configuración del sistema, y accesible por el usuario administrador. También existen dos puntos de entrada que son los primeros que se muestran cuando un usuario accede al sistema, y que son el ‘Panel principal’ y ‘Tareas pendientes’. Estos puntos de acceso son diferentes porque en el primero de ellos se muestra un panel resumen de cada tipo de ticket, y en ‘Tareas

pendientes' se muestra un resumen de las tareas del motor de BPM que tiene pendientes el usuario conectado.

5.6.2 Perfiles

Los perfiles equivalen a los roles dentro del sistema. Estos roles son asignados a los usuarios cuando se registran en la aplicación, y determinan las bandejas que pueden ver y acceder. En ITSM se distinguen los siguientes perfiles de usuario: **Admin** (el administrador del sistema), **Help Desk** (técnicos con permisos sobre todo tipo de ticket), **IT** (técnicos con permiso sobre los tickets en los que está asignado su grupo), **Requester** (el perfil del usuario solicitante, limitado a sus propios tickets).

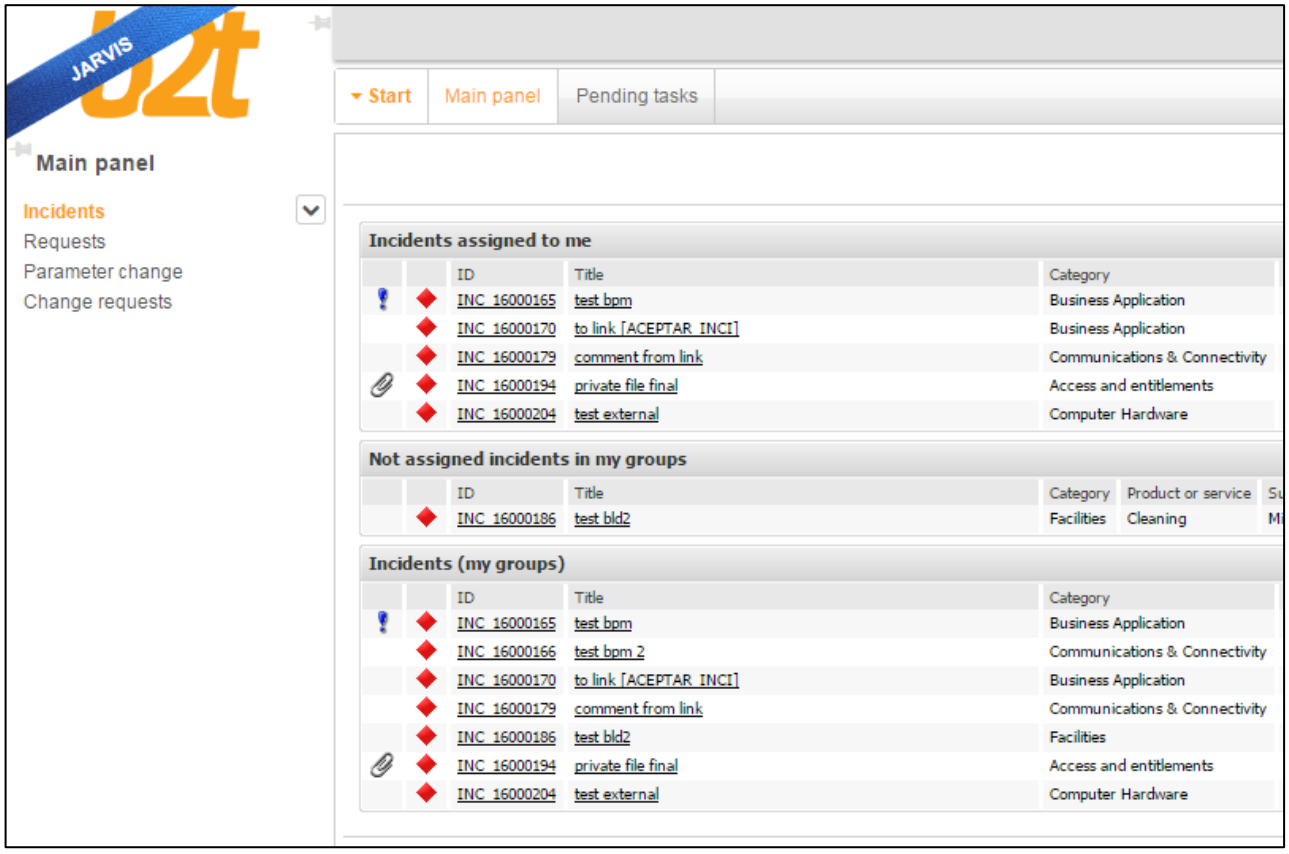

Figura 7: Pantalla principal de un técnico

6 Desarrollo

A lo largo de este capítulo se expondrán de forma detallada las implementaciones realizadas para solventar algunos de los requisitos definidos con anterioridad. Dado que el número de requisitos es extremadamente grande sólo se mostrarán los casos más representativos o que han supuesto mayor dificultad en su resolución. Para cada uno de los requisitos implementados el ciclo de desarrollo que se ha realizado consiste, en su mayor medida, en los siguientes pasos:

1. Ante un nuevo requisito, se identifica la parte del modelo a la que afecta, en concreto qué concepto o conceptos van a verse afectados. Puede ser una modificación de un concepto existente (como ocurre en el caso de las Incidencias) o la creación de nuevos conceptos (véase los nuevos conceptos de Cambios, Cambio de parámetros o Petición, y sus relacionados).
2. Se propone y diseña una solución, y se realizan las modificaciones del modelo necesarias: generalmente los conceptos, el modelo de datos y las interfaces.
3. Se generan los modelos modificados para actualizar los ficheros de clases y funciones. Todos los cambios efectuados en los modelos toman forma mediante el código generado por ScooP.
4. En este punto se realiza el apartado más técnico, ya que es cuando se realiza la mayor parte del código y se definen los algoritmos. Se codificarán todas las funciones necesarias para resolver la lógica establecida por el modelo.
5. Por último, se lanzaría el servicio en un entorno local para realizar las pruebas, primero de forma unitaria para los nuevos elementos, y después de forma general en el sistema para validar de forma completa.

[GEN-001] Asignación de permisos a los grupos resolutores.

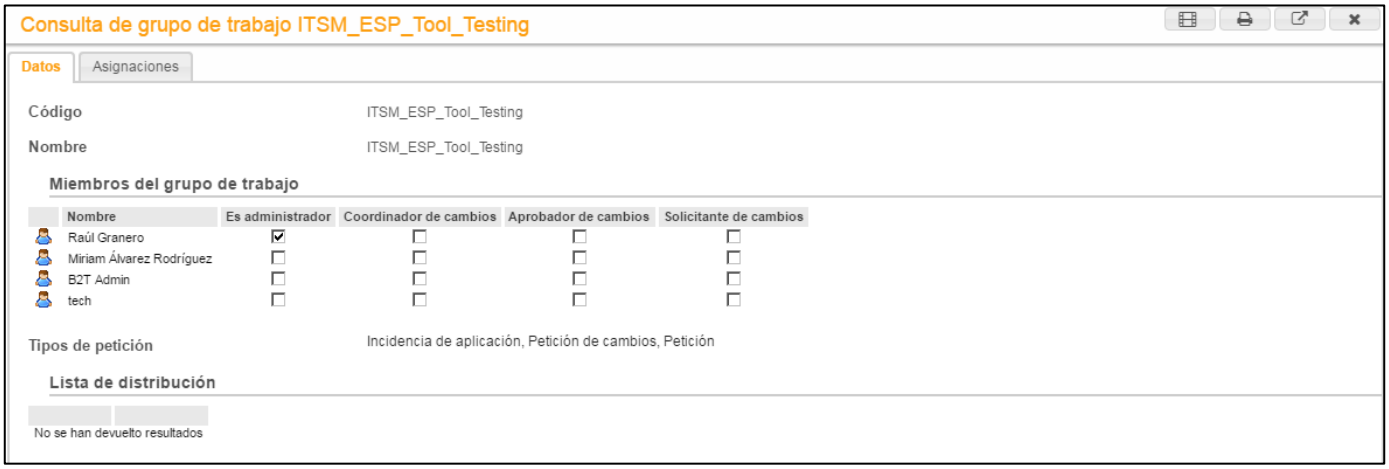

Figura 8: Consulta de un grupo de trabajo

Para la implementación de este requisito hemos tenido que crear una nueva tabla ITS_GRUPO_TIPOREQ que relaciona los grupos de trabajo de la aplicación con un tipo de solicitud. De esta forma, y mediante un control multicheck desde la interfaz, podemos asociar permisos para gestionar los diferentes tickets a los grupos resolutores.

Además, hemos modificado el concepto MiembroGrupoITSM para que soporte distintos roles dentro de un ticket de cambio. Estos roles se asignan mediante atributos de tipo flag que se han añadido al concepto base existente previamente.

[INC-CONS-001] Acciones en consulta: cambiar prioridad, tomar decisión, añadir comentarios + [INC-BPM-002] Asignar a *helpdesk* cuando la prioridad es Alta/Crítica.

Se quiere añadir acciones en la consulta del ticket, pero estas acciones sólo deben estar disponibles para los técnicos, y además sólo cuando se pueden utilizar (es decir, en caso de que una incidencia esté cerrada, por ejemplo, hay que controlar que no aparezcan los botones posibilitando acciones).

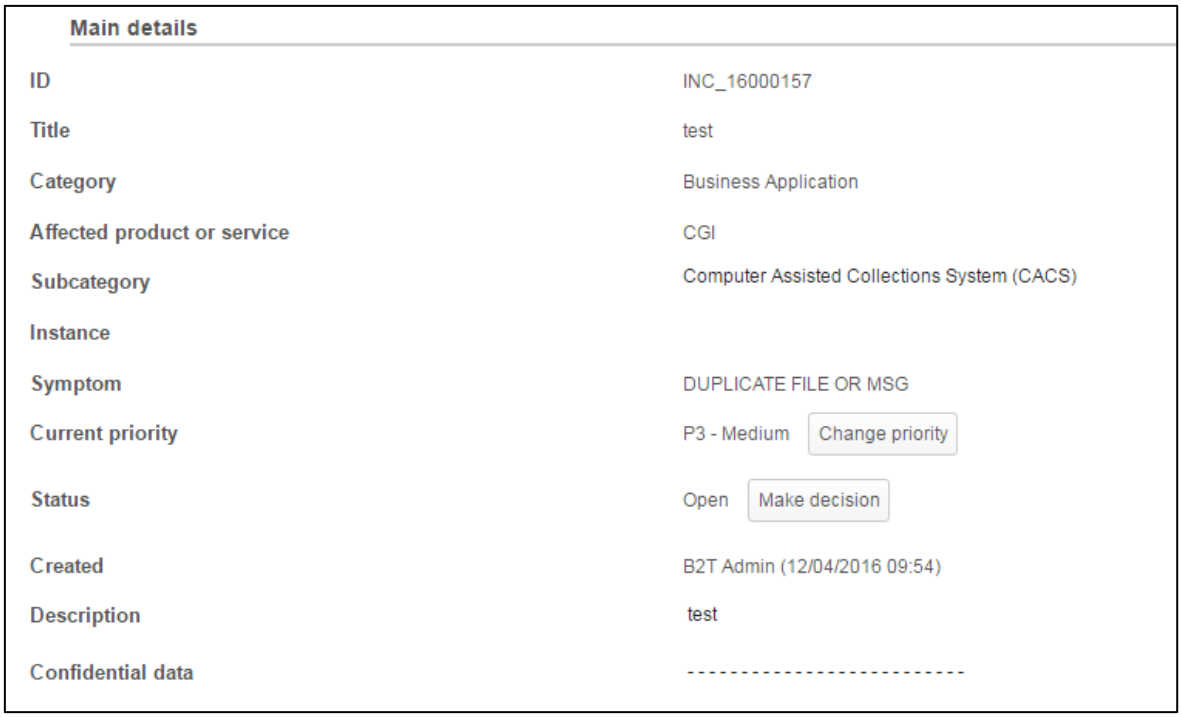

Figura 9: Consulta de una incidencia

Para establecer si los botones de 'Cambio de prioridad' y 'Toma de decisión' deben aparecer, primero es necesario definir el visor como dinámico. Esto significa que nuestra interfaz va a estar dividida en visores y controles constantes, y formaremos el visor global concatenando estos elementos. Cuando se define un control como constante, pasa a estar definido en el fichero de clases como si se tratara de un elemento de interfaz individual. Los visores agrupan controles, y los controles constantes se definen además de forma independiente. Pare definir el visor dinámico, utilizamos la función SetVsView, que construye el visor aplicando las restricciones establecidas por las banderas. Estas banderas comprueban mediante llamadas a funciones si un usuario tiene privilegios avanzados (es un técnico), si pertenece al grupo asociado al ticket, si el ticket se encuentra en un estado 'activo', o si es el usuario conectado el que consulta el ticket. Mediante estas banderas podemos establecer cuando aparecen los botones de forma dinámica.

Función SetVsView

```
public function SetVsView()
{
  var itemCat;
  var advancedFl = SessionUserHasAdvancedPriv();
  var groupFl = AdvancedPrivAndWorkGroup();
  var finalizedFl = (D_STATE in ["CIERRE","FINALIZADA","CANCELADA","ESPERA"]);
  var requesterFl = (GetSessionDataField("USR_CD") == REQUEST_USR);

  VS_DIN_VIEW = "";
  VS_DIN_VIEW += VS_VIEW_REQUESTER;
  VS_DIN_VIEW += VS_VIEW_MAIN;
  // Mostrar Criticidad de modulo solo para tecnicos
  VS_DIN_VIEW += (advancedFl) ? VS_VIEW_APP_ADV : VS_VIEW_APP;
  // Mostrar botones Change priority y Make decision solo si es Help desk, ADMIN o IT
asignado
  VS_DIN_VIEW += (IsChangePriorityAllowed()) ? VS_VIEW_STATE2 : VS_VIEW_STATE;
  if(IsSessionUserConfDataAllowed()) VS_DIN_VIEW += CTRL_DATOS_CONFID_DS;
  VS_DIN_VIEW += VS_VIEW_RESOL;
  if(D_STATE in ["CIERRE","FINALIZADA"])
  {
     itemCat = GetObjAppCategory();
     VS_DIN_VIEW += VS_VIEW_RESOL2;
     if(itemCat && (itemCat.APP_CATEGORY_TYPE_CD == "BLD"))
     {
        VS_DIN_VIEW += CTRL_RESOL_PRL_FL;
     }
  }

  VS_DOCUMENTS_DIN = "";
  if(groupFl || requesterFl) VS_DOCUMENTS_DIN += TOOLBAR_DOCUMENTS;
  var arrayDoc = new
FileDocumentAdHoc(this,"IS_LAST_DOCUMENT_VERSION_FROM_CONCEPTOBASE_AND_SUFFIX");
  if(!arrayDoc.IsEmpty())
  {
     VS_DOCUMENTS_DIN += CTRL_LISTA_FILEDOCUMENT_DATA_NULL;
  }
  //Related Records
  VS_RELATED_REC_DIN = "";
  if(groupFl && !(D_STATE in ["CANCELADA","ESPERA"])) VS_RELATED_REC_DIN +=
TOOLBAR_RELATED_RECORDS;
  VS_RELATED_REC_DIN += VS_RELATED_REC;
}
```

La función de cambiar prioridad recupera la prioridad actual del ticket y el nuevo valor escogido por el usuario. Sólo se produce el cambio si la prioridad nueva es diferente, y está justificado. Después de hacer estas validaciones, se produce el cambio de prioridad, y la modificación al flujo de BPM en caso de que la nueva prioridad sea crítica. Después de actualizar el concepto con los nuevos valores, se añade una entrada en el journal mediante la función CreateLogEntry, que crea una entrada de log. Para finalizar, en caso de que se haya escalado la incidencia a una prioridad crítica, se manda un email al grupo de HelpDesk/Centro de mando informando de que tiene asignada una nueva incidencia que tiene que tratar.

Función ValPGChangePrioridadFin

```
public function ValPGChangePrioridadFin()
{
  var objNewPrior = new PriorityItsm(PRIORIDAD_NEW_CD);
  var escalatedFl;

  /**Como llamamos a OnUpdate y esa función hace Reload(), se pierde información como
PRIORITY_DS o COMMENT_LOG_DS.
  Por eso almacenamos en variables esos valores antes de actualizar el concepto.*/
  var oldPriorityDs = PRIORITY_DS;
  var commentLogDs = COMMENT_LOG_DS;

  //Si es P1 o p2 se asigna a Command Center
  escalatedFl = AsignarCommandCenter(objNewPrior);

  PRIORITY_ID = PRIORIDAD_NEW_CD;
  SetLimiteDate(SysDate());
  OnUpdate();
  logComments =
"\\vITSM_REQINCIDENT_VALPGCHANGEPRIORIDADFIN_PSSS".Format(oldPriorityDs,objNewPrior.PRIO
RITY_DS,commentLogDs);
  CreateLogEntry("CHNG_PRIOR");
  Reload();
  //El evento de email se envía después para mandar la información actualizada
  if(escalatedFl) VariablesEventos.CreateObjEventAdHoc(this,"INC_EMAIL_COM_CENTER");
  ViewItem();
}
```

[INC-CONS-002] Enlace masivo de incidencias.

Para realizar el enlace masivo de incidencias, desde la consulta de una incidencia podremos acceder a la interfaz que nos permite ligar incidencias a la incidencia actual.

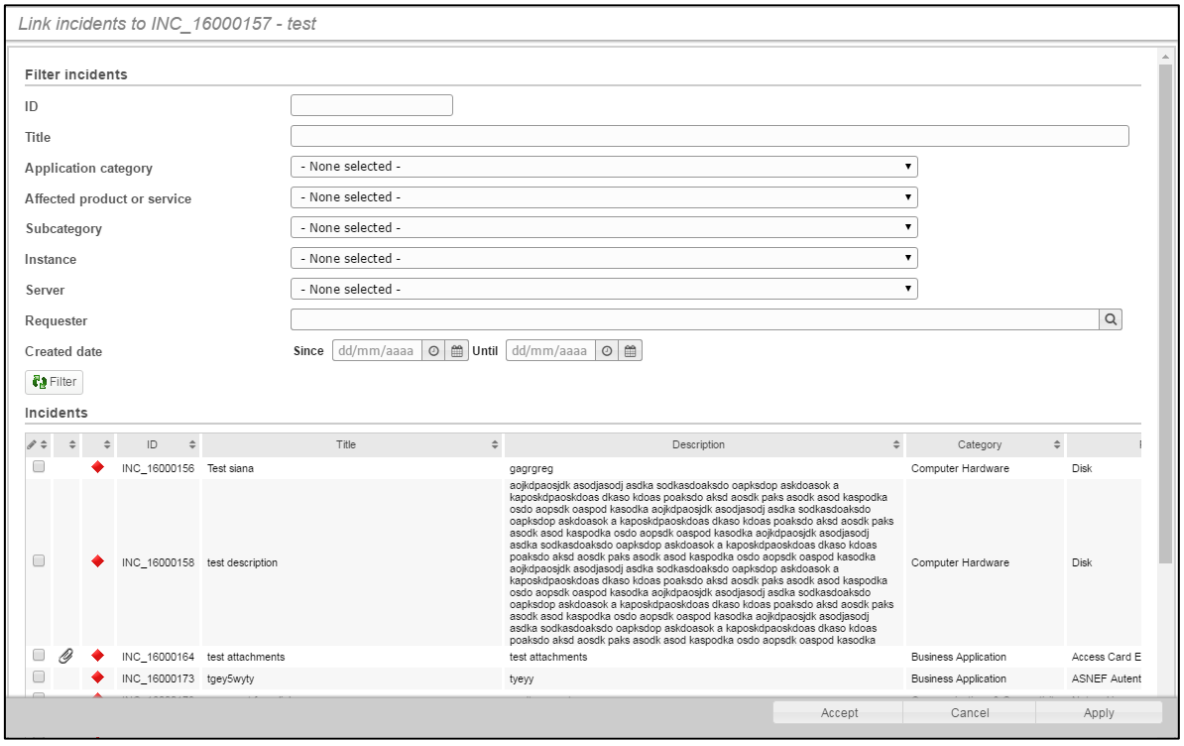

Figura 10: Formulario de link masivo de incidencias

Esta interfaz presenta en su parte superior un buscador, que permitirá filtrar las incidencias en el listado inferior según los criterios mostrados. Esto se consigue mediante una captura de los atributos definidos en los controles de la interfaz, y aplicando un constructor de búsqueda a la lista inferior a través del botón (filter). Para ello definimos la siguiente función.

Función SetBuilderFilterMassLink

```
public function SetBuilderFilterMassLink()
{
  var condStr = GetDataDefault(GetObjClassId(this),"REL") +
GetDataDefault(GetObjClassId(this),"SEL");

  condStr += " WHERE S.SERV_REQUEST_ID <> #num,SERV_REQUEST_ID; AND S.MAIN_REQUEST_ID
IS NULL";
  condStr += " AND S.IBPME_TAREA_ACTUAL_CD IN ('CATALOGAR','DIAGNOSTICO',
'DIAGNOSTICO_REOPEN', 'ACEPTAR_INCI')";
  condStr += " AND S.ESTADO_CD NOT IN ('CANCELADA',
'RECHAZADA','ESPERA','CIERRE','FINALIZADA')";

  condStr += AddFilterConditions();

  SetStaticData(GetObjClassId(this),"BUILD_SEARCH",condStr);
}
```

Esta función lo que permite es establecer las condiciones para que el constructor BUILD_SEARCH ejecute la consulta siguiendo los criterios establecidos por el usuario. La función AddFilterConditions es otra función en la que se encapsula los posibles valores que añadir a las condiciones, siendo estas la categoría de aplicación, subcategoría, etc.
Una vez filtradas y seleccionadas las incidencias de la lista, el botón aceptar de la página es el encargado de realizar la lógica de enlace mediante la función MakeMassLink, que recorre las incidencias seleccionadas e invoca a la función AsociarIncidencia.

Función MakeMassLink

```
public function MakeMassLink()
{
  var objIncident;

  for(key in GlobalesItsm.mapIncidents)
  {
       objIncident = new ReqIncident(key);
       objIncident.MAIN_REQUEST_ID = SERV_REQUEST_ID;
       if(objIncident.IBPME_TAREA_ACTUAL_CD in ["CATALOGAR", "DIAGNOSTICO",
"DIAGNOSTICO_REOPEN", "ACEPTAR_INCI"])
       {
            objIncident.MAIN_REQUEST_ID = SERV_REQUEST_ID;
            if (!objIncident.AsociarIncidencia()) return false;
            VariablesGlobalesIbpme.TomarDecisionBpm(objIncident, "ITS_INCIDENCIA",
"REPETITIVE_INCIDENT", GetSessionDataField("USR_CD"), "Massive link");
       }
  }
}

public function AsociarIncidencia()
{
  var objReq, arrayLinked, warnMsg, mainReqDs;
  if(Empty(MAIN_REQUEST_ID)) return false;
  objReq = new ReqIncident(MAIN_REQUEST_ID);
  if(!empty(objReq.MAIN_REQUEST_ID))
  {
       MAIN_REQUEST_ID = objReq.MAIN_REQUEST_ID;
       mainReqDs = objReq.MAIN_REQUEST_DS;
  }
```

```
  else mainReqDs = objReq.REQUEST_DS;
  OnUpdate();
  arrayLinked = new ReqIncident(this,"BUILD_FROM_MAIN_REQUEST_ID");
  for(objInc in arrayLinked)
  {
       objInc.ChangeParentIncFromAsociar(MAIN_REQUEST_ID);
  }

  logComments = "\\vITSM_REQINCIDENT_VALPGASOCIARINCI %s".Format(mainReqDs);
  CreateLogEntry("MIXBPMNEXT");
  return true;
}
```

Hay que tener en cuenta que asociar las incidencias implica modificar el flujo. Es por ello que en la función MakeMassLink se produce una llamada a la función de variables globales TomarDecisionBpm, que avanza el flujo de cada incidencia hasta la fase de 'Incidencia repetitiva'.

[INC-CONS-003] Desvincular una incidencia.

Para desvincular una incidencia, vaciamos el campo MAIN_REQUEST_ID que es el que contiene la referencia al padre, y actualizamos el concepto mediante la función OnUpdate(). Esta función lo que hace es actualizar los valores de base de datos con la información de la que dispone actualmente el concepto, y evalúa el modelo de estados (por si los nuevos datos actualizados produjeran una transición). Además, restablecemos el proceso de la incidencia hija para que vuelva a estar en progreso en vez de en espera de la incidencia relacionada. La función que lo permite es:

Función ContinueUnLinkIncident

```
public function ContinueUnlinkIncident()
{
      var requestDs;

      requestDs = MAIN_REQUEST_DS;
      MAIN_REQUEST_ID = "";
      OnUpdate();
      ApiBpme.TomarDecisionBpm(this, "ITS_INCIDENCIA", "ESPERA_MAIN_FINKO",
GetSessionDataField("USR_CD"),
"\\vITSM_REQINCIDENT_CONTINUEUNLINKINCIDENT_PS1".Format(requestDs));
      ViewItem();
}
```

[INC-EM-001] Poder añadir comentarios desde el email.

Se solicita poder añadir comentarios desde el email que se ha generado de forma automática desde la aplicación. Para ello, haremos uso de una técnica que genera un link externo a la aplicación. Mediante ese link, el usuario accederá a un formulario donde podrá añadir la información en forma de texto libre y *submitir* ese contenido directamente al sistema, sin acceder en ningún momento al mismo. Esto actualizaría el ticket añadiendo una entrada de log con el comentario introducido.

Para poder generar el link, hacemos uso de una función de librería llamada GetExternalLink.

```
linkDs = FunctionsProject.GetExternalLink(this, "AddCommentFromMail",REQUEST_USR);
```

Esta función recibe como argumento una nueva función, que compondrá el html que se mostrará cuando se pulse el link desde el email. Esta página se forma mediante un xsl

definido en los directorios del sistema. Obtendremos el xsl y lo transformaremos a html para mostrar la página donde el usuario podrá introducir el comentario.

Función ComponerPaginaAddCommentFromMail

```
public function ComponerPaginaAddCommentFromMail()
{
  var htmlDs, pathDs, xslFileDs, objXmlDoc;
  var objOpcionSel;

  ADDCOMMENT_HREF_DS =
FunctionsProject.GetExternalLink(this,"ValPgAddCommentFromMail",REQUEST_USR);

  htmlDs = "";
  objXmlDoc = new XML_DOC();
  pathDs = Ini_Read("Directories","XslFiles");
  xslFileDs = Path_Make("",pathDs,"addCommentFromMail","xsl");
  if(C3Existe(xslFileDs))
  {
       ObjToXml("DRAWXMLEX_ADDCOMMENT");
       objXmlDoc.LoadXml(XML);
       objXmlDoc.TransformNode(xslFileDs);
       htmlDs = objXmlDoc.M_TRANSFORMNODE;
       htmlDs = htmlDs.Translate();
  }
  else
  {
       System.out.format("No existe el fichero %s. No se puede realizar  la consulta
de proceso.", xslFileDs);
  }
  return htmlDs;
}
```

Una vez compuesta la página, y el usuario haya completado la información, para enviar los datos de vuelta al sistema es necesario primero realizar una validación de que el control de comentarios no está vacío, y en caso contrario, insertar el comentario introducido por el usuario. Si no hay ningún error durante el proceso se mostrará un mensaje html de confirmación al usuario indicando que se ha almacenado la información satisfactoriamente.

Función ValPgAddCommentFromMail

```
public function ValPgAddCommentFromMail(var mapParams)
{
  var comentariosDs;
  var usrCd;

  comentariosDs = mapParams["COMENTARIOS_DS"];
  usrCd = GetSessionDataField("USR_CD");

  if(empty(Trim(comentariosDs)))
  {
       System.out.write("Error en ValPgAddCommentFromMail. Error al guardar el
comentario.");
       //throw "Por favor introduzca un comentario";
       throw "\\vITSM_REQINCIDENT_VALPGADDCOMMENTFROMMAIL_1";
  }

  InsertCommentFromMail(comentariosDs,usrCd);

  <@
       <html>
            <head/>
            <body>
               <div>
               <table style="margin-left: auto; margin-right: auto; margin-top:
```

```
100px;">
                    <tr>
                         <td width='30px' height='30px'><img
src="/images/OKGreen30.gif" /></td>
                         <td><div><p style="font-weight: bold;">El comentario ha sido
registrado correctamente</p></div></td>
                    </tr>
                </table>
            </div>
            </body>
        </html>
  @>

  return state.Translate();
}
```

Función InsertCommentFromMail

```
public function InsertCommentFromMail(var commentDs, var usrCd)
{
  var objNew = new LogItsm();
  objNew.OnInitNewItem(this);
  objNew.B2TLOG_CD = "COMMENT_ITSM";
  objNew.B2TLOG_DS = commentDs;
  objNew.LOG_USR = usrCd;
  objNew.OnInsert();
}
/**Función GetEmailAsuntoCommandCenter*/
public function GetEmailAsuntoCommandCenter()
{
  var asuntoDs = "%s -
\\vITSM_REQINCIDENT_GETEMAILASUNTOCOMMANDCENTER_PS".Format(REQUEST_DS, PRIORITY_DS);
  asuntoDs = asuntoDs.Translate("SPANISH");
  return asuntoDs;
}
```

[CHG-CONS-001] Visibilidad bandeja y permiso de alta.

Para establecer esta restricción hacemos uso de los modelos de seguridad y los grupos de trabajo del sistema. Sólo tendrán acceso al punto de acceso del ticket de cambio aquellos grupos que tengan activa la marca de cambios en su definición de grupos (permisos establecidos en GEN-001). Además, para dar de alta, tienes que ser un miembro perteneciente a un grupo definido en el directorio activo de Windows. Las funciones que hemos definido para resolver este problema tienen la particularidad de que se ejecutan en el inicio de sesión del usuario, es decir, cuando un usuario hace login en el sistema. De esta forma, nada más entrar podemos establecer en su modelo de seguridad de forma dinámica si tiene acceso o no a las bandejas.

Función SetDataFromUsr

```
public function SetDataFromUsr(object usr)
{
  (...)

  CheckAltaChgPermission();
  CheckViewChgPermission(usr);
}
```

La función CheckAltaChgPermission es la que comprueba si el usuario que se conecta al sistema pertenece al grupo del directorio activo que tiene permiso para dar de alta un ticket de cambios. Para ello, recupera los grupos de AD (Active Directory) del usuario y los evalúa contra el grupo predefinido (este grupo es modificable por un administrador desde

la sección de configuración del sistema). Si se produce coincidencia, el usuario conectado añade a su lista de permisos el de creación de cambios.

Función CheckAltaChgPermission

```
public function CheckAltaChgPermission()
{
  var arrayGroupsAD, grupoChangeDs;

  if(FunctionsProject.UsuarioTienePrivilegio("ITSM_ADMIN"))
  {
       List_AddTail(this,"MENULIST","ALTA_CHG");
  }
  else
  {
       // Array con los nombres de los grupos de AD
       arrayGroupsAD                                                                    =
ActiveDirectory.getArraySegGroupsByLogin(#ini("Audit","Domain"),GetSessionDataField("EXT
_CD"));
       grupoChangeDs = B2tParams.GetParam("ITSM","PERMISOS_ALTA","Change");

       if(!arrayGroupsAD.IsEmpty() && !empty(grupoChangeDs))
       {
            //Comprobamos si el grupo que tiene permiso de alta está entre mis grupos de
AD
            for(groupDs in arrayGroupsAD)
            {
               //Si hay coincidencia asignamos el grupo de seguridad
               if(grupoChangeDs == groupDs) List_AddTail(this,"MENULIST","ALTA_CHG");
            }
       }
  }
}
```

La función CheckViewChgPermission comprueba los grupos de trabajo del sistema a los que pertenece el usuario conectado, y si alguno de ellos tiene permiso para gestionar tickets de cambios. Para ello, se obtiene el array de grupos mediante el constructor BUILD_FROM_USR_CD_TIPO_REQUEST_CAMBIOS y si el array devuelto no está vacío añadimos el permiso de consulta al usuario.

Función CheckViewChgPermission

```
public function CheckViewChgPermission(object usr)
{
  var arrayGrupos;

  if(FunctionsProject.UsuarioTieneAlgunPrivilegio(["ITSM_ADMIN","ALTA_CHG"]))
  {
       List_AddTail(this,"MENULIST","VIEW_CHG");
  }
  else
  {
       //Construimos los grupos de trabajo del usuario que tienen permiso de cambios
       arrayGrupos                                    =                                 new
GrupoTrabajoItsm(usr,"BUILD_FROM_USR_CD_TIPO_REQUEST_CAMBIOS");
       if(!arrayGrupos.IsEmpty())
       {
            List_AddTail(this,"MENULIST","VIEW_CHG");
       }
  }
}
```

Ambas funciones comprueban primero si el usuario conectado es un administrador mediante la llamada a la función UsuarioTieneAlgunPrivilegio, que comprueba si en el perfil del usuario conectado se encuentra el grupo de seguridad ITSM_ADMIN.

[CHG-BPM-003] Poder definir tareas, grupos aprobadores y grupos revisores.
Durante el proceso del cambio, en la fase de análisis la tarea que tiene el solicitante es definir qué tareas se realizarán para llevar a cabo el cambio, así como incluir grupos aprobadores que deberán su conformidad y, si es necesario, grupos revisores para mantenerlos informados de todo lo que suceda con el ticket.

Figura 11: Consulta en el cambio de las tareas y grupos

Para ello es necesario definir una interfaz especifica que permita dar de alta tareas, y asociar grupo de trabajo mediante un buscador y una acción de añadir elementos. Para añadir una nueva tarea, creamos un nuevo objeto TaskITSM y navegamos a su formulario de alta mediante la librería de Navigate.

Función AddChangeTask

```
public function AddChangeTask()
{
  var objChangeTask;

  objChangeTask = new TaskItsm();
  objChangeTask.OnInitNewItem(this);

  Navigate.NextPage(objChangeTask, "PG_ALTA");
}
```

El formulario de la tarea mantiene el formato establecido hasta el momento en el que se muestra en la parte superior la información del cambio del que proviene, y a continuación la información de la tarea propiamente dicha. Al finalizar el registro se produce una validación de los datos.

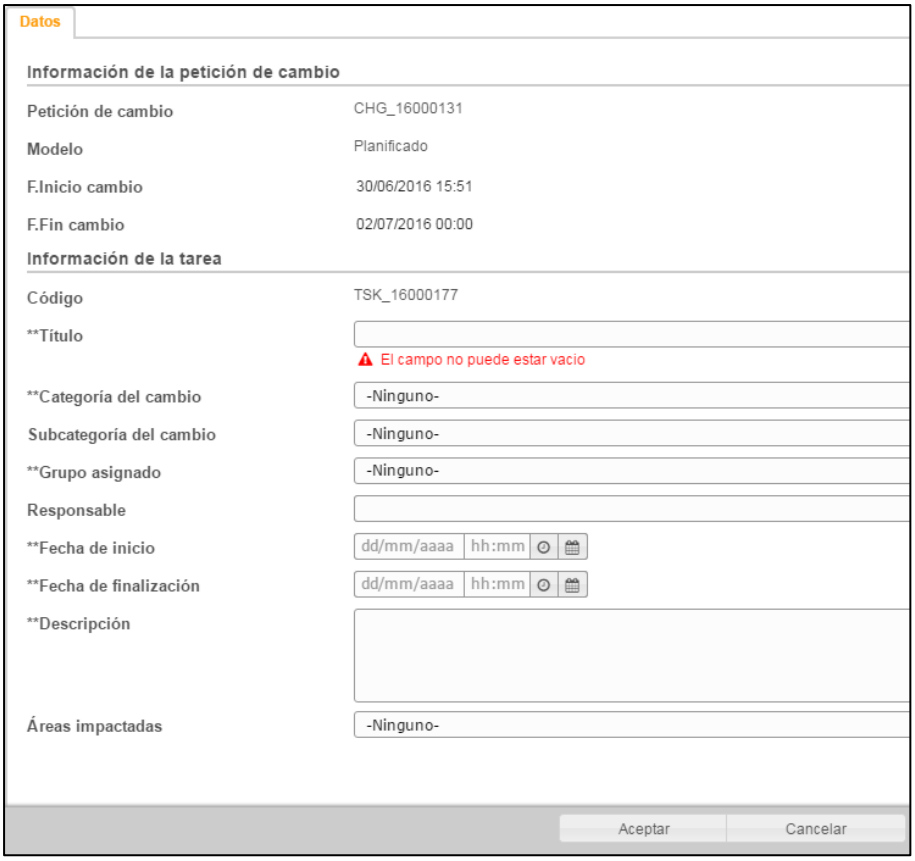

Figura 12: Formulario de alta de una tarea

Para poder añadir grupos aprobadores, se estableció como aproximación el poder buscar el grupo que te interesara y después asociarlo al ticket. Esto añade un registro en la tabla ITS_RELGRUPOCHG con el tipo de grupo aprobador (APR). El control de búsqueda permite introducir un texto en el control por el que facilitar la consulta y filtrar mejor los resultados. Así, será posible introducir el nombre, o parte del mismo para realizar la búsqueda. Este tipo de controles siempre lleva asociado un constructor del estilo BUILD_FROM_TIPO_REQUEST_CD_APPROVALGROUP_DS.

Constructor BUILD_FROM_TIPO_REQUEST_CD_APPROVALGROUP_DS

```
constant BUILD_FROM_TIPO_REQUEST_CD_APPROVALGROUP_DS =
  "@bind" + "@const,THIS_FROM;" + "@const,WHERE_COND;" +
  "@const,WHERE_AND; G.BORRADO_FL <> 1 AND G.GRUPO_TRABAJO_ID IN (SELECT" +
  " T.GRUPO_TRABAJO_ID FROM ITS_GRUPO_TIPOREQ T WHERE T.TIPO_REQUEST_CD = #data" +
  ",TIPO_REQUEST_CD;) AND G.GRUPO_TRABAJO_DS #like,APPROVALGROUP_DS; " +
  "@const,GROUP_BY;" +
  "@const,HAVING;" +
  "ORDER BY G.GRUPO_TRABAJO_DS " +
```

Para añadir al grupo aprobador, primero comprobaremos que se ha seleccionado algún elemento, de lo contrario no podríamos añadir nada y se mostraría un mensaje de error. También se verifica que no exista el grupo ya asociado al ticket como aprobador, para lo que se comprueba si existen objetos de tipo relación con el constructor BUILD_FROM_APPROVALGROUP_ID. En caso de que las validaciones sean correctas, crearemos un nuevo objeto RelGrupoChange, le asociaremos la información relevante del cambio y lo insertaremos en base de datos mediante la función OnInsert.

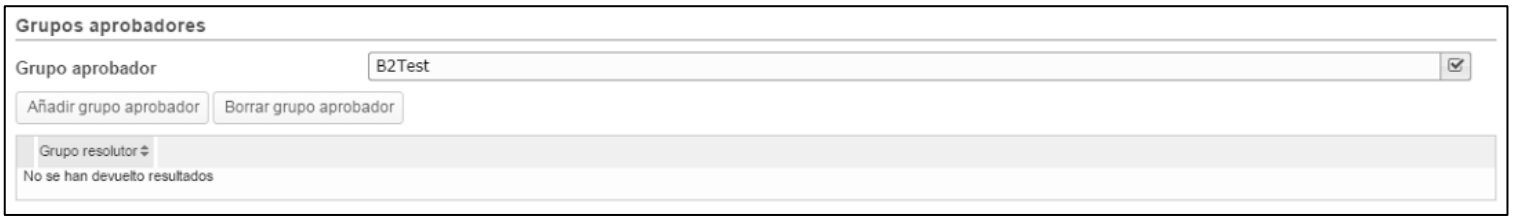

Figura 13: Interfaz de asociación de grupos aprobadores

Función AddGroupApproval

```
public function AddGroupApproval()
{
  var objRelGrupoChange;

  if(empty(APPROVALGROUP_ID))
  {
       iBetMessage("\\vIBET_ATENCION",  "\\vITSM_REQCHANGE_ADDGROUPAPPROVAL_1");
       return false;
  }
  //Comprobacion de que no se introduce el mismo grupo 2 veces
  if(ExistsObjects("RelGrupoChange", "BUILD_FROM_APPROVALGROUP_ID", this))
  {
       iBetMessage("\\vIBET_ATENCION",  "\\vITSM_REQCHANGE_ADDGROUPAPPROVAL_2");
       return false;
  }
  objRelGrupoChange = new RelGrupoChange();
  objRelGrupoChange.OnInitNewItem(this);
  objRelGrupoChange.GRUPO_TRABAJO_ID = APPROVALGROUP_ID;
  objRelGrupoChange.REQCHANGE_ID = REQCHANGE_ID;
  objRelGrupoChange.GROUP_TYPE_CD = "APR";
  objRelGrupoChange.OnInsert();

  //Vaciamos el campo y reabrimos
  APPROVALGROUP_ID = "";
  APPROVALGROUP_DS = "";
  Navigate.ReOpen();
}
```

Para los grupos revisores el procedimiento es análogo salgo que el GROUP_TYPE_CD sería “REV”, diferenciando en la misma tabla los dos tipos de grupos posibles.

7 Integración, pruebas y resultados

Una vez acabada la fase de implementación de los requisitos, las nuevas funcionalidades se prueban de forma unitaria, comprobando que los resultados son correctos. Después de se verifica el correcto comportamiento de las funcionalidades añadidas al sistema completo, y se comienza un ciclo de pruebas del sistema de forma global.

Las pruebas se realizan en un entorno de desarrollo con una instancia del servicio instalada de forma local. Cuando se codifica una nueva función, o se modifica algún modelo es necesario validar esos cambios en el sistema. El proceso de *testing* se puede dividir en las siguientes partes:

- Pruebas de **caja blanca**: se prueba que el procedimiento definido por el nuevo código no produzca ningún fallo. Para ello es necesario definir uno o varios casos de prueba que ejecuten los diferentes flujos de la función, y comprobar que se producen sin errores. En resumen, se verifica que las funciones no produzcan errores durante su ejecución.
- Pruebas de **caja negra**: una vez comprobado que no se producen errores durante la ejecución de la función, es necesario comprobar que se devuelve los valores esperados como resultado de las operaciones. Estas pruebas se centran en introducir datos al sistema para obtener resultados después de ejecutar las nuevas funciones añadidas. Con ellas se valida que las funciones devuelven los resultados correctos.
- Pruebas **unitarias**: las pruebas de caja blanca y negra se centran en funciones concretas o en nuevos elementos del modelo (como pueden ser las interfaces), pero es necesario verificar que la inclusión de estos nuevos elementos no provoque inconsistencias con el resto del módulo. Las pruebas unitarias se centran en probar el correcto funcionamiento del concepto, que en el caso de este sistema serían los tickets.
- Pruebas de **integración y sistema**: después de las pruebas unitarias con cada módulo, se realizan las pruebas generales, que se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez.

Para realizar las pruebas hacemos uso de la herramienta de desarrollo DEVR que dispone de *debugger* y de un fichero *core.out* donde se vuelcan todas las trazas. En caso de producirse algún error durante la ejecución de las pruebas es posible visualizarlo rápidamente en ese fichero, que además dispone de acceso directo a la función que ha producido el error y proporciona información sobre la línea en la que se produce y el tipo de error que es.

Después del proceso de pruebas interno, el sistema se instaló en un entorno de preproducción. Esta fase se denomina U.A.T (User Acceptance Testing) y es cuando el cliente tiene acceso por primera vez al sistema en un estado funcional. Durante esta fase, se produce un intercambio de *feedback* con las impresiones del cliente sobre usabilidad, estilo de interfaces, y resolución de las necesidades/requisitos especificadas.

8 Conclusiones y trabajo futuro

8.1 Conclusiones

Con respecto al trabajo realizado, durante el desarrollo de este proyecto se han analizado y llevado a cabo las tareas necesarias solicitadas por el cliente para mejorar el sistema de ticketing ITSM con los nuevos módulos. El sistema se encuentra instalado en el entorno de Producción del cliente y está siendo utilizado diariamente, llegando a ser la aplicación más utilizada en el día a día de la empresa, por lo que se deduce que se han conseguido los objetivos establecidos por el conjunto de los requisitos y se ha proporcionado un sistema estable y útil.

En relación a la ingeniería de modelos, he apreciado un cambio notable con las metodologías de trabajo estudiadas. La disponibilidad de herramientas que permiten la manipulación de modelos y la generación automática de código aumenta considerablemente la productividad, y disminuye el tiempo que se emplea programando, ya que si los modelos se diseñan correctamente y reflejan la realidad del negocio sólo habría que añadir funcionalidades de navegación o algún algoritmo de cálculos específicos. Por supuesto esta metodología tiene sus fallos si alguna de las partes (cliente-desarrolladores) no tiene suficiente conocimiento del proceso sobre el que va a realizar el modelo, lo cual puede llevar a la generación de proyectos "envenenados" que van a iterar y evolucionar hasta que sean insostenibles.

Los beneficios obtenidos durante el desarrollo de este proyecto son variados. Cabe destacar:

- La oportunidad de desarrollar un proyecto siguiendo el ciclo de vida completo del mismo, interfiriendo en la toma de decisiones y en las reuniones con el cliente.

- He podido consolidar conocimientos adquiridos durante los estudios de grado acerca del desarrollo de sistemas de información, programación orientada a objetos (los cuales me han resultado de utilidad para abstraer el desarrollo en modelos). También ha utilizado el conocimiento adquirido en la carrera sobre servidores y bases de datos, además del uso de patrones y algoritmos aprendidos en la misma.

- He podido formar parte de un equipo de desarrollo real, lo que me ha posibilitado aprender a utilizar las herramientas de organización y compartición de código (como por ejemplo *Subversion*) así como los elementos necesarios para poner en marcha un servicio orientado a la nube. También ha aprendido cómo es el proceso de desarrollo llevado a cabo por una empresa y participado activamente en el mismo.

8.2 Trabajo futuro

Como acabo de mencionar, el sistema de ticketing ITSM descrito en este documento ha pasado a un entorno de producción, abierto a los usuarios. Dado que es un sistema de gestión de información de gran magnitud, el acceso se está haciendo de forma gradual y comprobando que la integración de todos los datos es correcta. El cliente dispone de un

contrato de soporte con la empresa y por tanto cuando los datos se hayan comprobado y el número de accesos al sistema sea mayor, se efectuarán operaciones de soporte similares a las explicadas en esta memoria, solicitando mejoras o resolución de incidencias en caso de haberlas. Estas tareas de mantenimiento se realizarán por el mismo equipo de trabajo por lo que se espera que sigamos desarrollando nuevas especificaciones.

En el momento de entrega de la memoria, hay previstos nuevos módulos, aunque no se dispone de las especificaciones y por tanto no se ha comenzado su desarrollo. Se espera implementar un nuevo tipo de ticket denominado *Problema*, con el que se pretende proporcionar un medio para reportar el mal funcionamiento de un aparato o equipo de trabajo. Además, el sistema de ticketing ITSM será parte de un sistema mayor, que integrará varios servicios y que unificará todas las herramientas de uso interno actuales en el cliente. Este sistema general se denomina **IT Governance** y contará con subsistemas de gestión de la demanda, reporting/informes, y el ya mencionado ticketing, entre otros. Proyectos que se llevarán a cabo siguiendo la metodología explicada a lo largo de este documento.

Referencias

[1] B2T Concept, “Speed to market: Model-Driven Business Applications”, *Article*, pp. 3-5, Enero 2015

[2] Douglas C. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering", *Computer*, vol.39, no. 2, pp. 25-31, February 2006

Glosario

- Acción: toda aquella funcionalidad definida en un concepto y que generalmente está asociada a un botón, permitiendo al usuario actuar sobre los distintos conceptos.

- Business Process Management (BPM): metodología corporativa cuyo objetivo es la optimización de los procesos de negocio de una organización, a través de la gestión de procesos que se deben diseñar, modelar, organizar y documentar de forma continua.

- Concepto: representación de un objeto de negocio. Modela los elementos de los sistemas, englobando la información completa del elemento, incluyendo los datos y la lógica del objeto.

- Constructor: especifica un conjunto de expresiones de valores de fila que se va a construir en una tabla. El constructor del valor de tabla se puede especificar en la cláusula VALUES de la instrucción INSERT o en la definición de una tabla derivada en la cláusula FROM.

- Control: se refiere a un elemento individual de la interfaz, que puede ser de diversos tipos, entre los que destacan el combo, el botón de búsqueda, *radiobutton*, *checkbox*…

- Grid: elemento de la interfaz con el que se representan las listas en un sistema. Se puede definir un grid a nivel de concepto, indicando las columnas y atributos que debe mostrar.

- Lenguaje de dominio específico (DSL): es un lenguaje de programación o lenguaje de especificación ejecutable que ofrece, a través de las anotaciones apropiadas y abstracciones, poder expresivo centrado en, y no limitado a, un dominio de problema particular.

- MDE: metodología de desarrollo de software que se centra en la creación y explotación de modelos de dominio.

- Modelo: representaciones abstractas de los conocimientos y actividades que rigen un dominio de aplicación particular.

- Ontología: entendimiento común y compartido de un dominio. Una ontología es una base de datos que describe los conceptos en el mundo o algún dominio, algunas de sus propiedades y cómo los conceptos se relacionan entre sí. [Weigand (1997)]

- Punto de acceso: elemento de entrada a un concepto del sistema. Cada concepto presente en la aplicación que sea accesible por un usuario dispone de un punto de acceso.

Anexo A